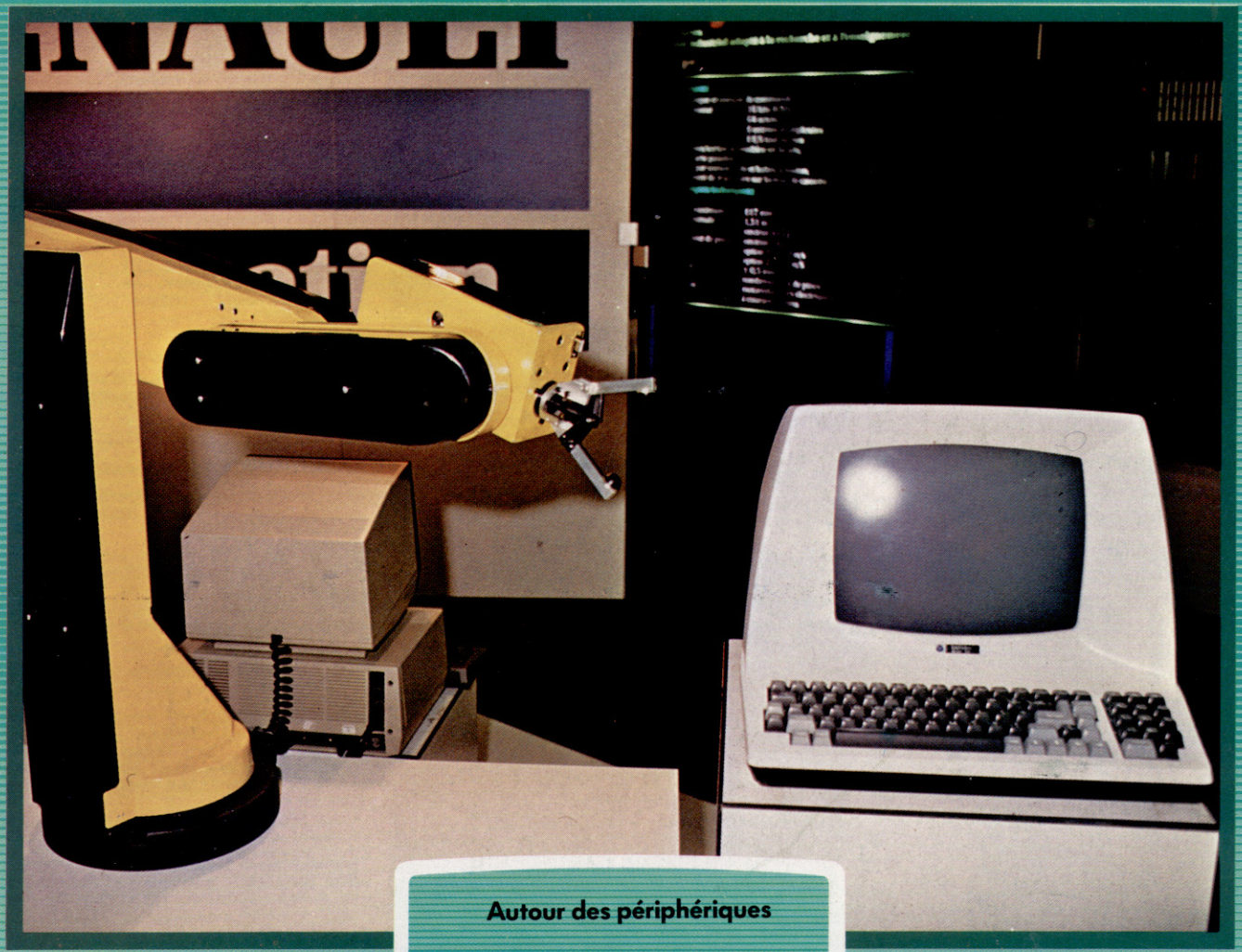
ABC INFORMATIQUE

N° 61

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Autour des périphériques

Ondes sonores

Le tableur Abacus

Logo et les chiffres

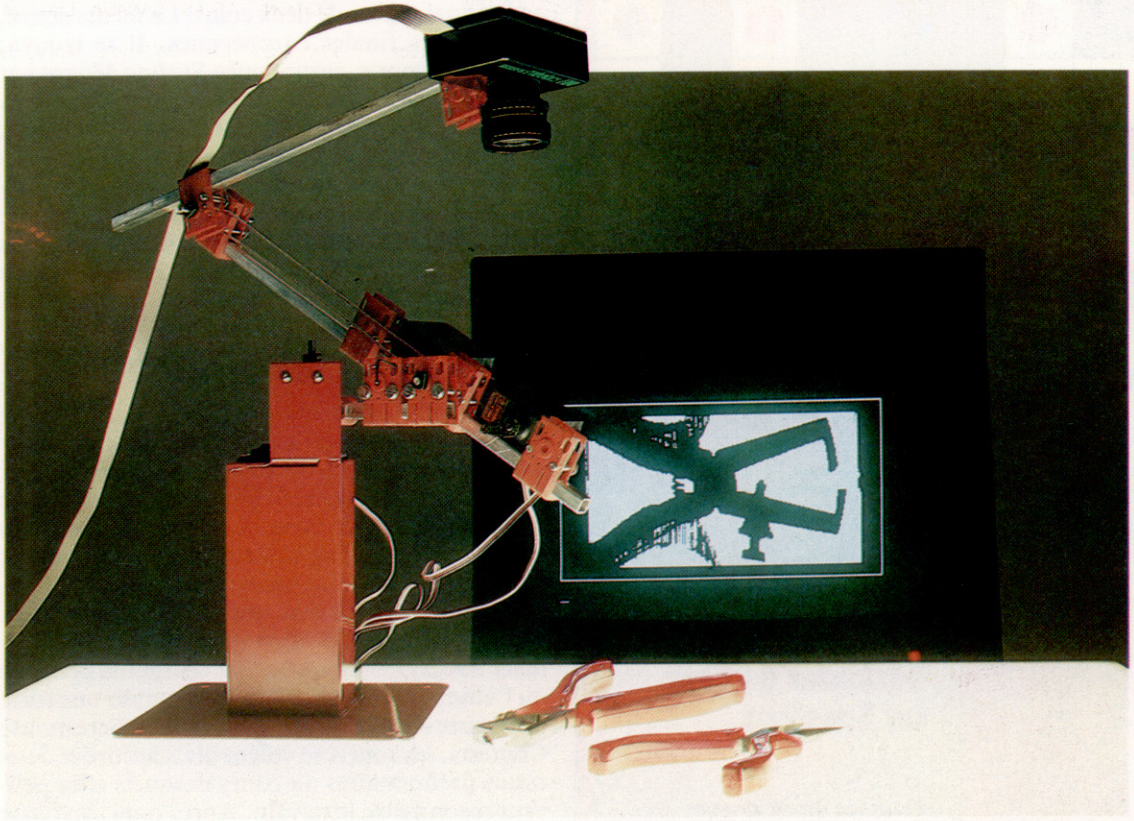
EDITIONS ATLAS

M 6062-61-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

Course d'obstacles

Un bras mécanique peut se comporter « intelligemment ». Nous allons voir ce qu'il faut faire pour construire un robot qui fasse réellement preuve d'« intelligence » en se déplaçant.



Apprendre à voir
Des robots mobiles sont capables d'apprendre un catalogue d'objets fondamentaux, qui seront ensuite traités par des algorithmes de reconnaissance des formes; il faut pour cela que ces appareils soient équipés de microprocesseurs évolués. Ici, un bras mécanique s'est vu adjoindre une caméra qui, à l'aide d'un logiciel spécialisé, permet d'obtenir des images numérisées. Une fois l'objet « vu » sous divers angles, le bras sera capable de le « reconnaître », quelle que soit sa position — ou, du moins, il aura de fortes chances de le faire. (Cl. Ian McKinnell.)

Avant toute chose, nous ne voulons pas d'un robot qui soit contrôlé par un opérateur humain. En effet, si ce dernier est obligé de l'observer sans cesse et de superviser ses moindres mouvements, autant lui faire accomplir directement toutes les tâches pour lesquelles l'appareil serait conçu! Bien entendu, cette règle ne s'applique pas dans toutes les situations. Lorsqu'il s'agit de manipuler des bombes, par exemple, mieux vaut laisser le robot s'en charger. Mais quoi qu'il arrive, la présence d'un homme chargé de le surveiller s'imposera.

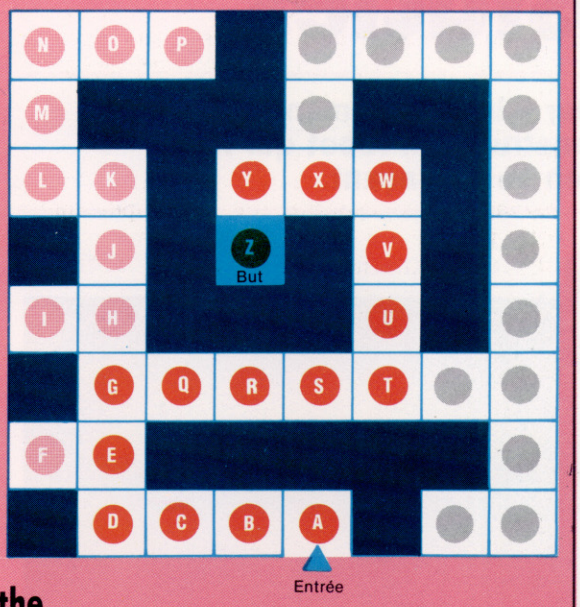
Il ne serait pas très utile non plus de commander l'appareil par le biais d'une série d'instructions stockées sur ordinateur. Ce ne serait plus qu'un simple automate qui exécuterait aveuglément un programme sans se soucier des circonstances particulières, même si là encore de tels procédés ont leur utilité.

Nous appellerons « robot intelligent » un robot qui, par exemple, est capable de vous apporter le café au lit le matin. De toute évidence, il n'est pas possible de le soumettre au contrôle d'un humain, puisqu'il doit mener à bien sa tâche avant que vous ne soyez réveillé! Et s'il obéit à une série d'instructions préprogrammées, il vaut mieux éviter de changer votre lit de place ou de laisser traîner des vêtements sur le plancher.

Dans de telles conditions, notre définition de l'intelligence dans un mouvement est la capacité que présente un robot à se déplacer dans un environnement sans être contrôlé par un être humain et sans suivre aveuglément une série de séquences fixes. Il devrait être capable de se déplacer d'un point à un autre en évitant tous les obstacles situés sur son chemin.

Les recherches relatives au domaine de l'intelligence artificielle ont souvent été entreprises. Certains jeux qui permettaient d'aborder des problèmes généraux beaucoup plus complexes ont été utilisés au cours des recherches se rapportant à l'intelligence artificielle : les programmes d'échecs par exemple. De même, les chercheurs se sont efforcés de mettre au point des robots capables de parcourir un labyrinthe. Les premiers concours opposant des amateurs passionnés sont apparus aux États-Unis vers la fin des années soixante-dix. Le principe de base est très simple : on construit un labyrinthe d'une surface de 3 m^2 environ, dans lequel des souris électroniques doivent se déplacer sans aucune aide pour parvenir le plus vite possible au centre. Le labyrinthe lui-

Simple comme bonjour
Des algorithmes très simples permettent de résoudre les problèmes de traversée d'un labyrinthe. Ce robot avance ici à travers des carrés vides jusqu'à ce qu'il se heurte à une impasse (ici les carrés F, I et P). Il fait alors machine arrière jusqu'au dernier carrefour qu'il rencontre (carré G) et considère désormais dépourvus d'intérêt tous les carrés qu'il a déjà franchis. S'il ne peut plus prendre un nouveau chemin à partir de ce carrefour, il remonte jusqu'au précédent, et ainsi de suite.
(Cl. Kevin Jones.)



Dans le labyrinthe

```
49 REM*****CBM 64**********
50 REM*  MAZE SOLVER    *
51 REM*****CBM 64**********
100 GOSUB 2000          :REM INITIALISATION
150 GOSUB 9000          :REM IMPRESSION LABYRINTHE
200 FOR L=1 TO 1
250 GOSUB 7000          :REM RECHERCHE
300 GOSUB 8000          :REM DEPLACEMENT
400 NEXT L
450 RO=16:CO=12:GOSUB 9500
500 IF MZ<>HM THEN PRINT"INSOLUBLE"
550 IF MZ=HM THEN PRINT"ET VOILA"
900 PRINT:PRINT:INPUT A$:RUN
1999 REM**********************
2000 REM*   INITIALISATION   *
2001 REM**********************
2100 SZ=10:S2=SZ*SZ:GX=SZ/2:GY=SZ+2
2120 DIM MZ(SZ,SZ),R$(4),LX(4),LY(4)
2140 X=RND(-TI)
2150 DEFFNR(N)=INT(RND(1)*N+1)
2160 HM=-1:GO=-2:WL= 42:W$=CHR$(WL)
2180 CL$=CHR$(147):H$=CHR$(19)
2200 X=GX-1:Y=GY-2:DR=3
2220 D$=CHR$(17):P$=D$
2240 FORK=1TO5:P$=P$+P$:NEXT K:P$=H$+P$
2400 DATA 0,1,"0",-1,0,".",0,-1,"",1,0,""
2420 FOR K=1 TO 4:READ LX(K),LY(K),R$(K)
2490 NEXT K:RETURN
6999 REM**********************
7000 REM*     RECHERCHE     *
7001 REM**********************
7110 RO=1:CO=1:GOSUB9500
7120 L=0:ND=0:FOR S=1 TO 4
7140 NX=X+LX(S):NY=Y+LY(S)
7200 IF NX<1 OR NX>SZ THEN NEXTS:RETURN
7220 IF NY<1 OR NY>SZ THEN NEXTS:RETURN
7230 MZ=MZ(NX,NY):IF MZ=0 THEN ND=S
7260 IF MZ=HM THEN ND=S:S=4:L=1
7280 NEXT S:RETURN
7999 REM**********************
8000 REM*     DEPLACEMENT     *
8001 REM**********************
8100 MZ(X,Y)=DR:FL=1
8120 RO=Y+1:CO=X+1:GOSUB 9500
8140 IF ND=0 THEN ND=DR-2-4*(DR<3):FL=2
8160 PRINT R$(FL);:X=X+LX(ND):Y=Y+LY(ND)
8180 DR=ND:IF Y>SZ THEN L=1:RETURN
8200 IF FL=2 THEN DR=MZ(X,Y)
8490 RETURN
8999 REM*****************************
9000 REM* IMPRESSION LABYRINTHE *
9001 REM*****************************
9040 PRINT CL$;:RO=1:CO=1
9050 WL=42:W$=CHR$(WL)
9060 S$="                    "
9070 FOR K=1 TO SZ+2:T$=T$+W$:NEXT K
9080 E$=W$+LEFT$(S$,SZ)+W$
9100 PRINT T$:FOR J=2 TO SZ+1
9120 PRINT E$:NEXT J:PRINT T$
9140 FOR K=1 TO S2/2
9150 WX=FNR(SZ):WY=FNR(SZ)
9160 MZ(WX,WY)=WL:RO=WY+1:CO=WX+1
9200 GOSUB 9500:PRINT W$:NEXT K
9250 CO=1+FNR(SZ):RO=1+FNR(SZ):GOSUB 9500
9300 PRINT"H";:MZ(CO-1,RO-1)=HM
9350 RO=GY:CO=GX:GOSUB9500:PRINT"^"
9490 RETURN
9499 REM**********************
9500 REM* POSITIONNECURSEUR  *
9501 REM**********************
9600 PRINT LEFT$(P$,RO)TAB(CO-1);:RETURN
```

Brian Morris

Variantes de basic sur Spectrum

Faire les changements suivants dans le programme :

```
49 REM*****SPECTRUM*********
50 REM*   MAZE SOLVER     *
51 REM*****SPECTRUM*********
2120 DIM MZ(SZ,SZ):DIM R$(4)
2130 DIM LX(4):DIM LY(4)
2140 RANDOMIZE
2150 DEFFNR(N)INT(RND*N+1)
9040 CLS:RO=1:CO=1
9600 PRINT AT (RO-1,CO-1);:RETURN
```

même est constitué de carrés de même superficie ; leurs côtés sont parfois ouverts (chemin possible) ou fermés (mur). La souris qui rejoint le centre du labyrinthe dans le temps le plus bref gagne le concours.

Lors du premier « Micromouse Contest » en Grande-Bretagne, il n'y eut que cinq participants. Les résultats ne furent guère probants : l'une des souris se révéla incapable d'aller en ligne droite, les autres ne savaient plus que faire dès qu'elles avaient passé plus de deux coins. La même année, il y eut des finales européennes. Il se trouva, enfin, une souris, surnommée *Stirling Mouse* par son inventeur, Nick Smith, pour parvenir au but. Elle était mue par un simple moteur pas à pas, et guidée par des capteurs mécaniques qui longeaient les parois du labyrinthe. Depuis, le succès de la compétition n'a fait que croître ; lors de la session 1984, tenue à Madrid, la souris la plus rapide mit 31,4 s pour gagner le centre. Presque toutes les autres purent y arriver.

Un plan du labyrinthe

Quels sont les problèmes posés par ce type d'appareil ? Une souris doit d'abord se déplacer avec suffisamment de précision pour pouvoir connaître à tout moment sa position exacte. Une méthode simple consiste à la monter sur roues et à la pourvoir d'un moteur pas à pas, souvent doté de dispositifs comme des codeurs de position angulaire qui assurent une certaine rétroaction positionnelle. Il est aussi nécessaire d'installer des capteurs capables de détecter la présence ou l'absence de murs, de façon à établir une sorte de « carte » du labyrinthe. Lors des Micromouse Contests, les souris se voient ainsi accorder deux essais préliminaires au cours desquels elles peuvent reconnaître le terrain. Après quoi, on passe à la compétition proprement dite, pendant laquelle on chronomètre les temps.

Les méthodes varient naturellement d'un constructeur à l'autre, mais une solution très classique consiste à équiper la souris d'un capteur tactile installé à l'avant. L'appareil se meut successivement jusqu'au centre de chacun des carrés du labyrinthe et vérifie s'il fait face ou non à un mur. Puis il effectue un déplacement de 90°, recommence l'opération, et ainsi de suite. Il finira ainsi par « connaître » l'emplacement exact des murs de chacun des carrés. L'information ainsi recueillie peut être contenue dans un seul nombre binaire de 4 bits — 1111, par exemple, indiquerait un carré bordé de quatre murs, ce qui n'est d'ailleurs pas concevable en pratique puisque la souris ne pourrait y pénétrer ! 0000 correspondrait à un carré dépourvu de tout mur, et 0111 à un carré avec trois murs et une seule ouverture, c'est-à-dire à un cul-de-sac.

Toutes ces données pourraient être stockées dans un tableau à deux entrées. C'est ainsi qu'en BASIC, DIM A(16,16) permettrait de représenter un labyrinthe de 16 « cases » de côté. La tâche de la souris consisterait alors à trouver un chemin qui la conduirait jusqu'à A(8,8), donc au centre. Elle est fréquemment pourvue d'un programme

qui étudie tous les déplacements possibles à partir d'une structure arborescente. Bien des branches de celle-là mènent à des impasses ou encore ramènent la souris à un endroit où elle est déjà passée; elles sont donc écartées. Le programme examine alors les possibilités restantes afin de déterminer le chemin le plus court jusqu'au centre, et c'est celui-là qui sera choisi. Cette méthode peut être adaptée pour fournir une stratégie plus efficace.

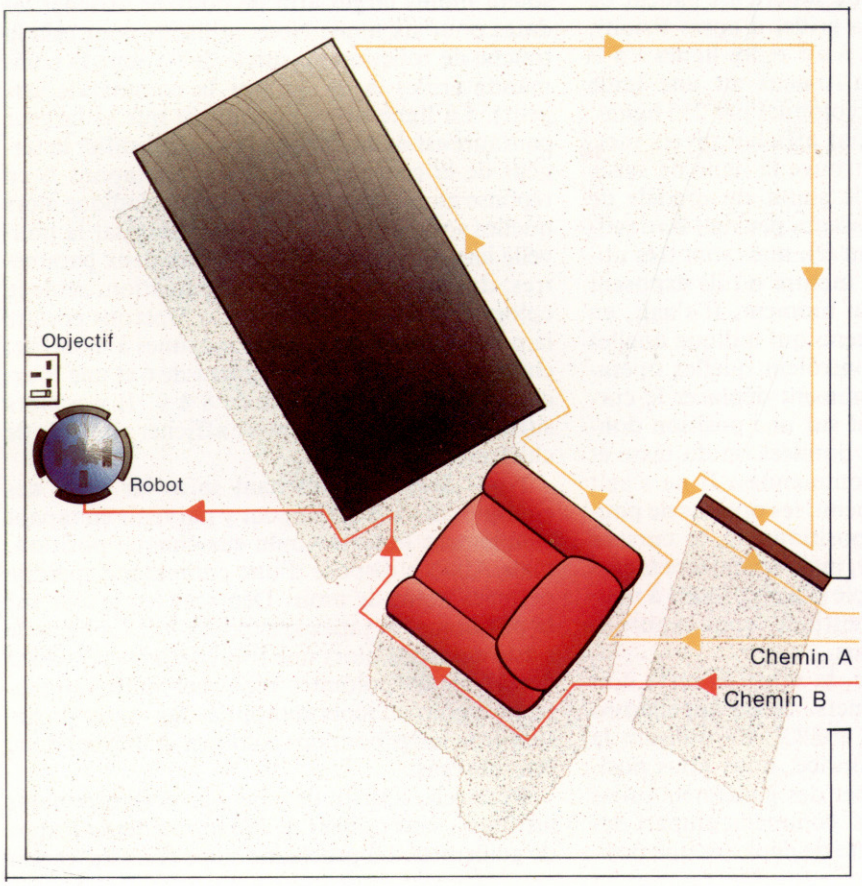
Les capteurs jouent un rôle essentiel. De simples détecteurs « tactiles » mécaniques imposent à la souris de buter dans chacun des murs avant de pouvoir dresser un plan; il est préférable de recourir à des dispositifs dits « de proximité », qui peuvent repérer les obstacles sans entrer en contact avec eux, ou à un capteur de distance — qui fait le même travail à partir de l'extrémité d'une ligne droite. De même quatre capteurs, au lieu d'un seul, pourraient « voir » simultanément dans les quatre directions, et ainsi la souris n'aurait plus à tourner sur elle-même en traversant chaque carré.

Dans la maison

Un robot parcourant un labyrinthe se conduit donc de manière « intelligente ». A bien des égards, les problèmes seront les mêmes quand on cherchera à en construire un qui puisse se déplacer dans votre appartement sans se perdre. Il lui faudra en effet faire usage de ses capteurs pour repérer la position de chacun des objets qui se trouvent dans une pièce, puis déterminer un chemin qui le mènera à sa destination en évitant les obstacles. Le gros problème est qu'une pièce est infiniment plus complexe qu'un labyrinthe. Elle n'est pas divisée en carrés, et les objets peuvent y changer de place : si une chaise est déplacée, ou si le chat se couche sur le tapis, le robot doit pouvoir en tenir compte. En outre, le robot est loin de connaître la psychologie des chats; on peut imaginer qu'il soit pris pour une souris, auquel cas on peut imaginer le pire pour son avenir!

Le seul moyen de résoudre la difficulté consiste à imposer à l'appareil un usage continu de ses capteurs, afin qu'il remette constamment à jour sa « carte » intérieure. La meilleure solution est sans doute d'équiper le robot d'un détecteur de mouvements, qui est un capteur capable d'enregistrer les variations de distance. Une fois qu'il a détecté un objet qui se déplace, le robot n'aura qu'à s'arrêter, jusqu'à ce que l'objet lui-même cesse également de se mouvoir, ou sorte de son champ de détection. Cela peut sembler absurde, et peut-être pourrait-on programmer l'appareil pour qu'il s'avance vers le chat et lui donne une caresse. Toutefois, il est à noter que de nombreux animaux s'immobilisent dès qu'ils ont perçu une présence étrangère.

Le problème du mouvement « intelligent » est donc lié par nature à l'emploi conjugué de capteurs gérés par un programme d'ordinateur. Plus ces capteurs seront nombreux, plus le robot parviendra à une bonne connaissance du monde qui l'entoure, et c'est cette connaissance, en fait, qui lui tient lieu d'intelligence véritable.



Une chambre à soi
Trouver un chemin parmi des objets inconnus n'est pas chose facile. Le chemin A montre le parcours accompli par un simple robot domestique s'efforçant de trouver une prise électrique. Il est guidé par un algorithme élémentaire qui consiste à longer les murs et les bords des objets, tandis que ses capteurs à courte distance examinent le tout. Le chemin B est celui qu'a suivi un robot du même genre, doté d'un algorithme un peu amélioré; lorsqu'il doit contourner un objet, il le fait toujours selon l'angle le plus fermé possible, ce qui lui épargne des recherches supplémentaires. Cette modification minime le rend beaucoup moins sensible aux erreurs et lui permet de faire un usage plus productif de ses capteurs. (Cl. Kevin Jones.)

Conduite modèle

Notre série d'articles sur les tableurs et modèles financiers a été jusqu'ici consacrée au progiciel Vu-Calc de Psion. Nous étudions ici Abacus, un tableur également de chez Psion destiné au QL Sinclair.

Menu des messages

Menu de commande

Une grande partie de l'intérêt suscité par le QL s'est reportée sur les quatre progiciels fournis avec la machine : Quill, le traitement de texte; Archive, la base de données; Easel, le programme graphique; Abacus, le tableur. Ces progiciels relèvent, pour certains éléments, de la conception logicielle intégrée. Ils peuvent s'échanger librement des données, et les modèles du tableur peuvent être affichés en tant que graphiques ou intégrés à un document préparé par Quill.

Il y a deux façons de charger Abacus (et les autres programmes) : la première consiste à introduire la cartouche Abacus dans le microlecteur 1 et à frapper la touche de fonction F1 pour sélectionner l'option moniteur, ou F2 si un téléviseur est utilisé comme mode d'affichage. Les progiciels QL comprennent des routines de chargement; les programmes se chargent donc automatiquement; la seconde s'applique au cas où l'écran est déjà choisi (en supposant qu'Abacus se trouve déjà dans le lecteur 1) et revient à taper lrun mdv 1_boot. L'écran initial d'Abacus apparaît alors.

L'écran affiche la partie supérieure gauche de la matrice du tableur (la grille, d'après Psion). Au départ, les colonnes A à F et les lignes 1 à 15 sont affichées, bien qu'Abacus ait une taille maximale de grille de 64 colonnes par 255 lignes. Au-dessus de la grille sont affichés divers messages, et au-dessous se trouve la ligne de saisie des données ainsi que le statut du modèle de tableau présent. Les messages peuvent être retirés de l'écran (en utilisant F2), mais sont très utiles au débutant dans la mesure où ils exposent les choix possibles à tout moment. Il s'agit, en réalité, d'une zone de menu qui indique quelles touches de fonctions contrôlent quelles opérations et qui explique comment déplacer le curseur ou aller directement sur une position donnée, comment saisir des données ou du texte et comment appeler des commandes. Il ne s'agit pourtant pas d'un vrai menu, les options ne pouvant être retenues en positionnant le curseur devant le choix approprié, mais devant être tapées par l'utilisateur. L'exemple suivant, fondé à nouveau sur un budget domestique, vous expose le fonctionnement d'Abacus.

Comme pour Vu-Calc, la saisie de texte doit être précédée de guillemets. Nous appellerons notre modèle PRÉVISION DE REVENUS. En utilisant la touche de curseur appropriée, nous nous positionnons sur D1 pour taper des guillemets suivis du texte du titre. Abacus, comme la plupart des tableurs, permet à un texte de dépasser une position si les positions voisines sont vides. Ainsi, il est facile de saisir même de longs titres en tout lieu de la grille.

Prévision des revenus

Nous pouvons également souligner un titre. A cette fin, il nous faut déplacer le curseur sur la position en dessous de la première position du titre (D2) et taper reproduiretitre(«=»,longueur[d1]). Reproduire est l'équivalent de la commande REPLICATE de Vu-Calc, et = indique au programme le symbole à utiliser (nous utilisons le signe = pour obtenir un double souligné). Le reste de la commande indique au programme de souligner toute la longueur du texte en D1, en répétant le signe choisi.

Contrairement à Vu-Calc qui a une largeur de colonne fixe de neuf caractères, Abacus nous permet de choisir différentes largeurs pour différentes applications. Nous avons besoin, pour notre exemple, d'une colonne A plus large, afin qu'elle puisse recevoir un texte de longueur variable. Il faut également ici que les colonnes restantes soient moins larges afin de pouvoir afficher les dates pour six mois. Nous utilisons pour cela la touche de fonction F3, suivie de G (pour la commande Grille) et de L (pour la commande Largeur). La ligne de saisie indique que la largeur courante est de 10. Nous voulons modifier la largeur de la colonne A en 15, nous tapons 15 en réponse au message. Le programme indique alors quelles sont les positions concernées par la nouvelle largeur. En tapant A pour les deux paramètres, de début et de fin de modification, seule la colonne A sera modifiée. Nous répétons ensuite la procédure pour les autres colonnes à modifier, choisissant cette fois une largeur de 6 et une fourchette de colonnes allant de B à G. Il y a alors suffisamment de place pour afficher six mois de chiffres.

Nous devons maintenant saisir les titres des colonnes des mois. On peut y parvenir en tapant simplement le texte voulu sur chaque position, mais Abacus dispose d'une commande spéciale pour les noms de mois. Déplacez le curseur sur la position A3 et tapez ligne=mois(col()-1). La ligne de saisie demande alors une fourchette — répondez B et G, et les colonnes reçoivent automatiquement le nom du mois approprié. Janvier et février sont trop longs pour nos colonnes et doivent donc être abrégés.

Pour cela, il suffit de taper par-dessus les noms attribués, sans oublier de faire précéder la frappe de guillemets. La prochaine étape consiste à saisir

les titres des lignes en déplaçant le curseur d'une ligne vers le bas après chaque saisie. Notre modèle financier suppose que le budget concerne un vendeur dont le revenu est constitué d'un salaire et de commissions. Les ventes effectuées peuvent être saisies au fur et à mesure des mois, et des prévisions adaptées peuvent alors être faites sur la suite de l'exercice. Les valeurs négatives sont affichées entre crochets dans la grille terminée. Voici les titres à faire figurer : « Ventes : estimation et résultats », « Commissions », « Salaire de base », « Revenus totaux », « Dépenses : emprunts, loyer, eau, électricité, gaz, téléphone », « Total des dépenses », « Solde net positif (ou négatif) », « Ouverture et fermeture de la balance bancaire ».

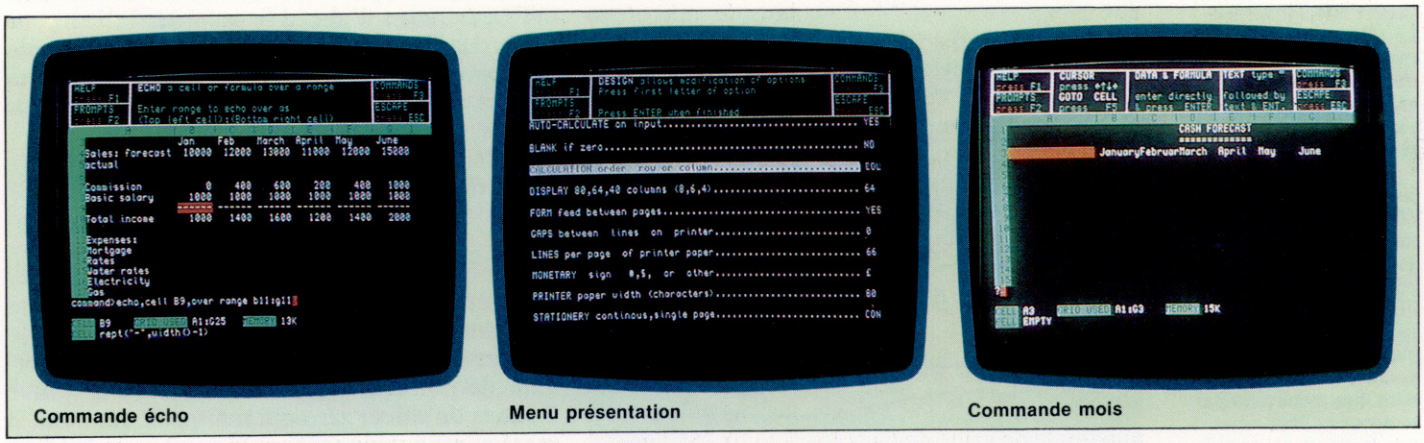
Une fois les titres des colonnes et des lignes saisis, nous pouvons commencer à entrer des chiffres. Nous saisirons d'abord les estimations de ventes pour les six mois affichés. Il s'agit de 10 000 F, 12 000 F, 13 000 F, 11 000 F, 12 000 F et 15 000 F. Les chiffres doivent être tapés sans le F de francs. Les ventes effectives ne peuvent pas encore être saisies, aussi continuerons-nous avec le calcul des commissions basées sur les estimations de ventes. Les commissions se calculent sur la base de 20 % du chiffre des ventes lorsqu'elles dépassent 10 000 F. La formule mathématique à faire figurer sur la position B10 est : ligne=(ventes-10000)*.2. A nouveau, la fourchette est de B à G. Répondez par ces lettres aux messages, et les commissions seront instantanément calculées et affichées.

Si le salaire de base est de 1 000 F par mois, on le fera figurer en position B11 par ligne=1000, sur la fourchette de mois B à G. La formule mathématique pour les revenus totaux doit être entrée en B13. Il s'agit de ligne=somme(col), avec une gamme de lignes allant de 0 à 11, et de colonnes de B à G. Le calcul du revenu est alors programmé, mais le modèle peut être amélioré en encadrant la ligne « Revenus totaux ». Déplacez le curseur en B12 et saisissez : ligne=reproduiretitre(« - »,largeur()-2). Abacus répondra en soulignant de quatre tirets chaque chiffre se rapportant au salaire de base. Pour effectuer la même opération sur la ligne 14, afin de mettre des tirets au-dessus et au-dessous de la ligne « Revenus totaux », utilisez la commande ÉCHO. Le curseur étant toujours en B9, le système demande sur quelle longueur faire le souligné. Répondez par : B14:G14. Si vous avez oublié de laisser des lignes d'avance à la saisie des titres, vous pouvez les insérer avec la commande GRID. Pour terminer le formatage, les soulignés doivent être justifiés à droite par la commande J sur la gamme B2 à G14. Les chiffres et les tirets seront alors alignés.

Les dépenses peuvent maintenant être saisies en utilisant la formule mathématique de ligne pour les chiffres mensuels appartenant à des catégories déterminées, telles que les emprunts, ou en tapant simplement chaque chiffre sur la position appropriée pour des dépenses comme le gaz ou l'électricité. Les dépenses totales peuvent alors être définies comme la somme de toutes les dépenses, de la même manière que pour les revenus totaux.

De manière similaire, le solde net, positif ou négatif, peut être calculé de la même façon que les revenus totaux moins les dépenses totales. Malheureusement, Abacus semble ignorer toute partie d'une commande qui suit un espace; nous ne pouvons donc pas taper ligne=Revenus totaux-Dépenses totales. Il faut donc rebaptiser la ligne des revenus totaux par « revenus », puis taper ligne=revenus-total à nouveau de B à G. Chaque rentrée et chaque sortie mensuelle nette apparaîtra ainsi à l'écran.

En dernier lieu, il faut calculer les équilibres bancaires. Tapez d'abord le solde de départ, un déficit de 200 F par exemple. Vous l'exprimerez par : -200. Calculez ensuite le solde de clôture par la formule mathématique ligne=net+ouverture, le curseur étant en B28 pour la gamme habituelle comprise entre B et G. Vous obtenez le solde de clôture pour janvier. L'ouverture des soldes pour les autres mois se calcule par la formule ligne=B28 en C27. Pour étendre ce système à toutes les colonnes, il faut d'abord changer l'ordre de calcul des lignes aux colonnes par la commande PRÉSENTATION. Tous les soldes d'ouverture et de fermeture seront alors calculés et recalculés immédiatement lorsqu'un chiffre sera modifié. Notre modèle affichera les soldes négatifs en les faisant précéder d'un signe moins. Pour modifier ce format de telle manière que les crochets remplacent le signe moins, utilisez la commande UNITÉ et répondez au message par B.



Au-delà des mots
Une bonne présentation d'écran, des commandes efficaces et un système d'aide et de messages complet font d'Abacus un tableur agréable et facile d'utilisation. On voit la commande ÉCHO pour recopier une ligne sur l'autre, la commande PRÉSENTATION pour formater le tableau et la commande MOIS, qui fait apparaître chaque mois sur simple pression d'une touche. On voit également l'écran par défaut avec le menu des messages en haut et le tableau vierge montrant le menu de commande. (Cl. Ian McKinnell.)

Quel est ton nom?

Comment modifier notre programme de recherche de variable de telle façon qu'il puisse modifier le nom de telle ou telle variable? Nous donnons ici deux versions.

Ce nouveau programme est plus complexe que celui que nous vous avions déjà présenté. C'est pourquoi nous aurons besoin du langage machine. Le microprocesseur du BBC Micro (6502) et celui du Commodore 64 (6510) ont le même langage d'assemblage, aussi pouvons-nous les étudier en même temps.

Il nous faudra d'abord mettre au point une méthode permettant de stocker simultanément deux programmes en mémoire. Comme nous l'avons déjà vu, il suffit sur le BBC Micro de modifier deux des variables BASIC, PAGE et HIMEM. Sur le Commodore 64, par contre, nous aurons besoin d'un programme en langage machine qui altérera les différents pointeurs de la page zéro de la mémoire. C'est ce que la première partie du listage en assembleur, commençant par le label SWITCH, fera pour nous.

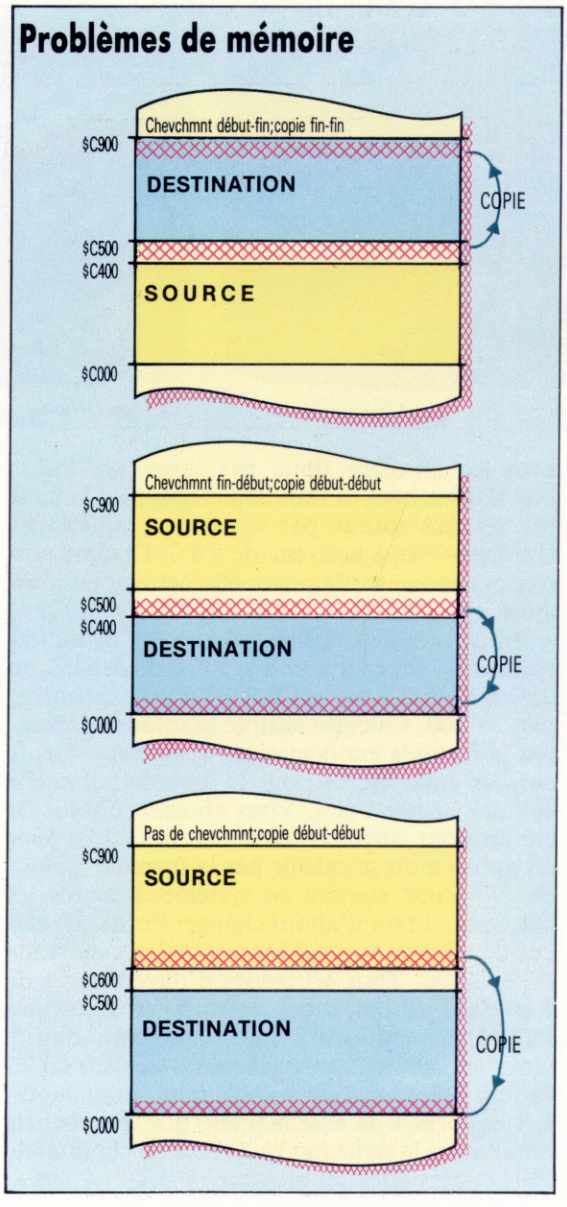
Nous pourrons ainsi manipuler deux programmes BASIC; l'un commençant à l'adresse $800 (emplacement habituel pour un programme BASIC), l'autre à l'adresse $9000 ($ signale que le chiffre qui suit est sous forme hexadécimale). SWITCH examine d'abord le pointeur TXTTAB pour voir celle des deux zones mémoire qui est prise en compte par l'ordinateur, puis modifie les valeurs contenues dans ce pointeur pour qu'il se tourne vers l'autre zone.

TXTTAB est donc altéré de façon à signaler le début du second programme. Après quoi, FRETOP et MEMSIZ doivent également pointer l'octet qui suit le dernier octet du nouvel espace mémoire, tandis que FRESPC prend en charge la fin de cet emplacement. Le programme recherche alors la série des pointeurs de chaîne d'adressage afin de trouver la fin du programme BASIC, VARTAB lui servant de pointeur temporaire. Lorsqu'il trouve 2 octets de valeur zéro qui marquent cette fin, il incrémente à deux reprises le pointeur précédent et copie le résultat en ARYTAB et STREND. Ceux-ci, tout comme VARTAB, pointent donc tous l'octet qui suit immédiatement le programme BASIC.

Les sous-programmes des lignes 30500 et 30600 constituent les principaux changements à effectuer par rapport au programme d'origine. Le premier trouve la fin du programme BASIC. Sur le BBC Micro, il fait usage des octets de longueur de ligne, et, sur le Commodore 64, des pointeurs indiquant la ligne suivante.

La seconde routine effectue les changements de noms de variables. Lorsque l'ancien et le nouveau noms ont le même nombre de lettres, le second est tout simplement écrit sur le premier. Dans le cas inverse, les choses sont un peu plus compliquées. Le programme doit en effet supprimer les espaces superflus, ou en ajouter d'autres, puis modifier en conséquence les variables dont il se sert pour garder trace de sa position au sein du programme en cours de modification. Il doit aussi faire des changements du même ordre sur l'octet de longueur de ligne (BBC Micro) ou sur les pointeurs indiquant la ligne suivante (Commodore 64).

Des sous-programmes en langage machine ajoutent ou suppriment les espaces. Cela, bien



Copier intelligemment
La routine de remplacement doit déplacer vers le haut ou le bas de la mémoire de vastes portions du programme BASIC, lorsqu'elle insère dans celui-là de nouveaux noms de variables de longueurs différentes. Elle doit alors tenir compte des adresses de départ et d'arrivée. Si elle recopie toujours en partant du début du bloc source, la fin de celui-là peut être recouverte par le début du bloc objet (de destination), ce qui entraîne la destruction d'une partie des données. Une routine de copie « intelligente » tiendra compte d'une telle possibilité et recopiera le bloc source en partant de la fin, contrairement à ce que fait une routine « mécanique ».
(Cl. Liz Dixon.)

sûr, pourrait être fait en BASIC, mais au prix d'une lenteur insupportable, puisqu'il faut, chaque fois, déplacer des milliers d'octets.

Ces sous-programmes sont au nombre de deux. UP ajoute un espace, DOWN en supprime un. Ils sont informés du bloc de programme à modifier par le biais des adresses mémoire CURR, LAST et DIFF.

Pour UP, voici comment il faut initialiser ces adresses :

CURR : adresse de la partie inférieure du bloc.
LAST : adresse du sommet du bloc, moins un.
DIFF: nombre d'octets à libérer.

Et pour DOWN :

CURR : adresse du sommet du bloc.
LAST : adresse de la partie inférieure du bloc, moins un.
DIFF : nombre d'octets à modifier.

Vous observerez que les deux sous-programmes partent chacun d'une des extrémités du bloc et se déplacent en sens opposé. Cela permet d'éviter l'effacement des données avant que les changements aient été effectués.

En réserve

Pour mettre en œuvre la version BBC du programme de remplacement de variables, tapez ou chargez d'abord le programme en assembleur, puis faites RUN afin de placer en mémoire le code machine. Chargez ensuite le programme à modifier et tapez :

```
P% = PAGE
```

Son adresse de départ sera ainsi transmise au programme modificateur, que vous chargerez et lancerez après avoir affecté à PAGE une valeur supérieure à LOMEM. Vous pouvez toujours revenir au programme de base avec :

```
HIMEM = PAGE - 1
PAGE = P%
```

Pour vous servir de la version Commodore du même programme, vous devrez également mettre en mémoire le code machine, soit à l'aide d'un chargeur BASIC, soit par assemblage du code source. Dans ce dernier cas, ou si vous chargez directement le code machine en tant que tel, il vous faudra POKEr la valeur zéro aux adresses 36864, 36865 et 36866. Cela revient à accomplir, sur la zone mémoire mise « en réserve », la commande NEW.

Cela étant fait, SYS 49152 vous permettra de passer d'une zone à l'autre. Si par hasard vous ne savez plus où vous en êtes, vous n'aurez qu'à faire en mode direct PRINT PEEK (44). 8 indique que vous êtes en zone « normale » et 144 en zone « en réserve ».

Une fois que vous disposez du code machine dans l'ordinateur, vous pouvez charger le programme à modifier dans la première zone, le programme de remplacement de variable dans la zone « réserve » du programme, puis faites RUN pour exécuter ce dernier.

Switch et Copie pour C64

```
10 ADRES=49152
20 FOR LIGNE=1000 TO 1180 STEP 10
30 S=0
40 FOR ADRES= ADRES TO ADRES+7
50 READ OCTET
60 POKE ADRES, OCTET
70 S=S+OCTET
80 NEXT ADRES
90 READ CHECKSUM
100 IF S<>CHECKSUM THEN PRINT"ERREUR DE
 DATA LIGNE"; LIGNE : END
110 NEXT LIGNE
120 POKE 36864,0 : POKE 34865, 0 : POKE 36866,0

1000 DATA162,0,164,44,192,8,208,9,787
1010 DATA160,160,32,72,192,169,144,208,1137
1020 DATA7,160,144,32,72,192,169,8,784
1030 DATA162,1,134,43,133,44,134,45,696
1040 DATA133,46,160,0,177,45,170,200,931
1050 DATA177,45,224,0,208,240,201,0,1095
1060 DATA208,236,230,45,208,2,230,46,1205
1070 DATA136,16,247,165,45,133,47,133,922
1080 DATA49,165,46,133,48,133,50,96,720
1090 DATA134,51,132,52,134,55,132,56,746
1100 DATA202,136,134,53,132,54,96,160,967
1110 DATA0,177,251,164,255,145,251,160,1403
1120 DATA0,196,251,208,2,198,252,198,1305
1130 DATA251,165,253,197,251,208,234,165,1724
1140 DATA254,197,252,208,228,96,164,255,1654
1150 DATA177,251,160,0,145,251,230,251,1465
1160 DATA208,2,230,252,165,253,197,251,1558
1170 DATA208,236,165,254,197,252,208,230,1750
1180 DATA96,0,0,0,0,0,0,0,96
```

Copie pour BBC

```
 10 MODE 7
 20 HIMEM=HIMEN-&100
 30 CURR = &70
 40 LAST = &74
 50 DIFF = &78
 60 FOR PASS = 1  TO 2
 70 P%=HIMEM
 80[ OPT 1
 90 .UP     LDY #0
100 .UP     LDA (CURR),Y
110         LDY DIFF
120         STA (CURR),Y
130         LDY #0
140         CPY CURR
150         BNE UP2
160         DEC CURR+1
170 .UP2    DEC CURR
180         LDA LAST
190         CMP CURR
200         BNE UP1
210         LDA LAST+1
220         CMP CURR+1
230         BNE UP1
240         RTS
250 .DOWN   LDY DIFF
260         LDA (CURR),Y
270         LDY #0
280         STA (CURR),Y
290         INC CURR
300         BNE DOWN1
310         INC CURR+1
320 .DOWN1  LDA LAST
330         CMP CURR
340         BNE DOWN
350         LDA LAST+1
360         CMP CURR+1
370         BNE DOWN
380         RTS
390]
400 NEXT PASS
410PRINT~UP,~DOWN
```

Remplacement de variable

Commodore 64
Faites subir au programme les modifications suivantes :

```
30005 INPUT «REMPLACER PAR»;R$
30006 CURR = 251
30007 LAST = 253
30008 DI = 255
30035 GOSUB 30500
30036 IF ERR THEN PRINT«FIN DU PROGRAMME
  INTROUVABLE» : END
30060 IF NXTLIGNE = 0 THEN END
30065 CURRLIGNE = PTRTEXT
```

Supprimez la ligne 30085

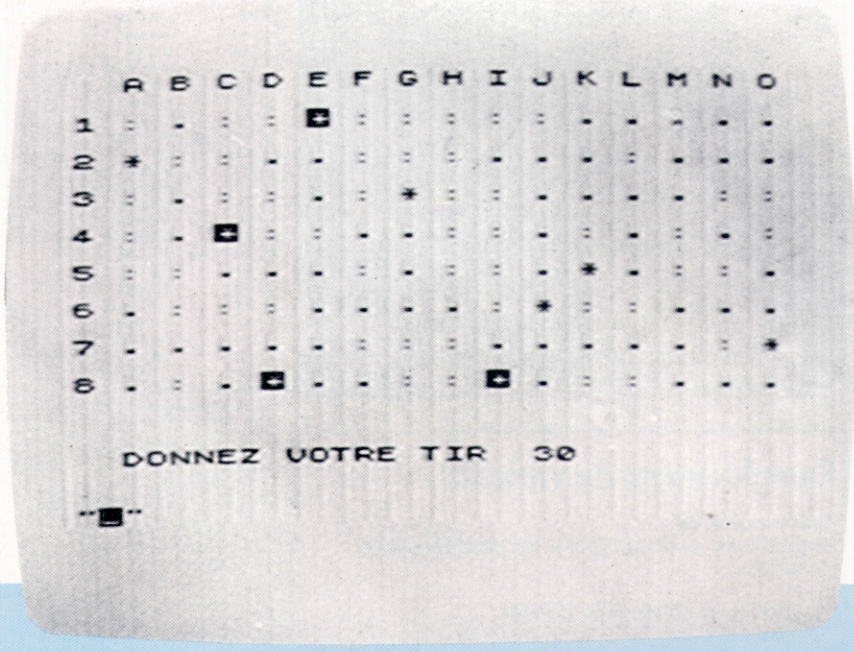
```
30460 IF NOM$ = T$ THEN GOSUB 30600
30465 IF ERR THEN PRINT«PAS DE PLACE LIGNE»;
LIGNENO : END
```

BBC Micro
Faites subir au programme les modifications suivantes :

```
30005 INPUT «Remplacer par»;REPLACEMENT$
30006 CURR = &70
30007 LAST = &74
30008 DIFF = &78
30055 GOSUB 30500
30056 IF Outofrom THEN PRINT «Fin du programme
  introuvable» : PRINT» PAGE au mauvais endroit?»:
  END
30060 Pointeurdetexte = P%
30070 IF Ligneno > 32767 THEN END
30105 Longueuroctet = Pointeurdetexte
30460 IF Nom$ = Cible$ THEN GOSUB 30600
30465 IF Outofrom THEN PRINT«Pas de place
  ligne»;Ligneno : END
```

Bataille navale - ZX81

Ce jeu de Mick Wilson vous propose une bataille navale contre votre ZX81. Il suit les règles du jeu habituel. L'extension mémoire de 16 K est peut-être utile.



Lorsque vous lancez le programme, la grille apparaît sur l'écran et l'ordinateur vous demande les positions de vos navires. L'ordinateur choisit ensuite les positions de ses propres navires. Après cette opération, le programme entre dans un cycle en demandant tour à tour les coordonnées des tirs de chacun des adversaires. Vos bateaux apparaissent sur l'écran sous la forme d'un astérisque, et les bateaux coulés sous celle d'un astérisque inversé. Les secteurs touchés apparaissent sous la forme du signe «:», et les secteurs non joués sous celle du signe «.». Le jeu se termine lorsque l'un des deux adversaires a perdu tous ses navires. Plus le jeu avance et plus l'ordinateur ralentit. Il est très important de bien suivre le listage en le tapant. En modifiant le nombre 7 dans les lignes 250, 330, 401, 460 et 552, vous pouvez modifier le nombre de bateaux. Une partie dure de 25 à 40 minutes.

```
  10 REM BATAILLE NAVALE
  20 REM SAUTER SOUS-PROGRAMMES
  30 GOTO 100
  40 LET X=(((CODE A$(1))-29)*2
)-1)*2
  50 LET Y=((((CODE A$(2))-38)*2
)-1)*2
  60 RETURN
  70 LET T=PEEK (PEEK 16398+256*
PEEK 16399)
  80 RETURN
 100 REM AFFICHAGE
 110 PRINT " ";
 120 FOR I=38 TO 52
 130 PRINT " ";CHR$ (I);
 140 NEXT I
 150 PRINT
 160 PRINT
 170 FOR I=29 TO 36
 180 PRINT CHR$ (I);" ";
 185 FOR J=1 TO 15
 190 PRINT " ";
 200 NEXT J
 204 PRINT
 205 PRINT
 210 NEXT I
 211 RAND
 220 PRINT AT 20,0;"PLACEZ VOS B
ATEAUX,"
 230 FOR K=1 TO 60
 240 NEXT K
 250 FOR I=1 TO 7
 260 PRINT AT 20,0;"COORDONNEES
DU BATEAU  ";I
 270 INPUT A$
 275 IF LEN A$<>2 THEN GOTO 270
 280 IF CODE A$(1)<29 OR CODE A$
(1)>36 OR CODE A$(2)<38 OR CODE
A$(2)>52 THEN GOTO 270
 290 GOSUB 40
 300 PRINT AT X,Y;
 302 GOSUB 70
 304 IF T=CODE "*" THEN GOTO 270
 305 PRINT "*"
 310 NEXT I
 320 PRINT AT 20,0;"JE CHOISI ME
S POSITIONS,,,"
 330 DIM S$(7,2)
 340 FOR I=1 TO 7
 350 LET A$=CHR$ (INT (RND*8)+29
)+CHR$ (INT (RND*14)+38)
 360 GOSUB 40
 365 PRINT AT X,Y;
 370 GOSUB 70
 380 IF T=CODE "*" THEN GOTO 350
 390 LET S$(I)=A$
 400 NEXT I
 401 FOR J=1 TO 7
 402 FOR K=1 TO 7
 403 IF J=K THEN GOTO 405
 404 IF S$(J)=S$(K) THEN GOTO 34
0
 405 NEXT K
 406 NEXT J
 410 LET M$=7
 420 LET Y$=7
 421 LET N=0
 425 LET N=N+1
 430 PRINT AT 20,0;"  DONNEZ VOT
RE TIR  ";N;"
 440 INPUT A$
 445 IF LEN A$<>2 THEN GOTO 440
 450 IF CODE A$(1)<29 OR CODE A$
(1)>36 OR CODE A$(2)<38 OR CODE
A$(2)>52 THEN GOTO 440
 460 FOR I=1 TO 7
 470 IF A$=S$(I) THEN GOTO 700
 480 NEXT I
 490 GOSUB 40
 500 PRINT AT X,Y;
 510 GOSUB 70
 520 IF T<>CODE "." THEN GOTO 44
0
 530 PRINT ":";AT 20,0;A$(1);" "
;A$(2);"      RATE.
 535 FOR I=1 TO 50
 536 NEXT I
 540 PRINT AT 20,0;"JE TIRE  ";
N;"
 550 LET A$=CHR$ (INT (RND*8)+29
)+CHR$ (INT (RND*14)+38)
 552 FOR H=1 TO 7
 553 IF A$=S$(H) THEN GOTO 550
 554 NEXT H
 560 GOSUB 40
 570 PRINT AT X,Y;
 580 GOSUB 70
 590 IF T=CODE ":" OR T=CODE "■"
 THEN GOTO 550
 600 IF T=CODE "*" THEN GOTO 650
 610 PRINT ":";AT 20,0;A$(1);" "
;A$(2);"      RATE.
 620 FOR I=1 TO 50
 630 NEXT I
 640 GOTO 425
 650 PRINT AT X,Y;"■"
 651 PRINT AT 20,0;"COULE.  IL V
OUS EN RESTE ";
 660 LET Y$=Y$-1
 670 PRINT Y$
 675 IF Y$=0 THEN GOTO 740
 680 FOR I=1 TO 100
 690 NEXT I
 695 GOTO 425
 700 GOSUB 40
 701 PRINT AT X,Y;"■"
 705 PRINT AT 20,0;"COULE.  IL M
EN RESTE ";
 710 LET M$=M$-1
 720 PRINT M$
 725 IF M$=0 THEN GOTO 740
 730 GOTO 675
 740 IF Y$=0 THEN LET W$="J AI"
 750 IF Y$=0 THEN LET L$="VOUS A
VEZ"
 760 IF M$=0 THEN LET W$="VOUS A
VEZ"
 770 IF M$=0 THEN LET L$="J AI"
 780 FOR I=1 TO 20
 800 NEXT I
 810 CLS
 820 PRINT W$;" GAGNE."
 830 PRINT
 840 PRINT L$;" PERDU."
 850 STOP
```

Regards périphériques

L'achat d'un ordinateur domestique n'est que la première étape de la réalisation d'un système complet avant d'adapter les périphériques offerts sur le marché.

Il y a peu de temps, le périphérique le plus important pour l'utilisateur d'ordinateur domestique était un module enfichable renfermant une mémoire supplémentaire. La mémoire étant coûteuse, des machines comme le ZX81 ou encore l'Alice étaient conçues de façon à garder des coûts de fabrication aussi bas que possible.

Les ordinateurs, aujourd'hui, ont jusqu'à 64 K de mémoire RAM intégrée (parfois plus); les modules RAM supplémentaires sont donc rarement nécessaires. L'utilisateur se voit aussi offrir un large choix de périphériques, allant de modems qui permettent la communication entre des utilisateurs éloignés par des centaines de kilomètres, des bras robots commandés en utilisant une interface adéquate, jusqu'aux synthétiseurs de parole qui trouvent des applications récréatives ou éducatives.

Bien qu'il y ait de nombreux dispositifs périphériques sur le marché, la plupart sont fabriqués uniquement pour les machines les plus courantes. Cela est un élément essentiel dont vous devrez tenir compte lors de l'achat d'un ordinateur pour avoir le meilleur choix.

Les nouvelles machines mettent du temps à constituer un marché de périphériques, mais on peut penser que la norme MSX récemment introduite simplifiera les choses en permettant d'utiliser des dispositifs complémentaires avec toutes les marques qui répondent aux caractéristiques MSX.

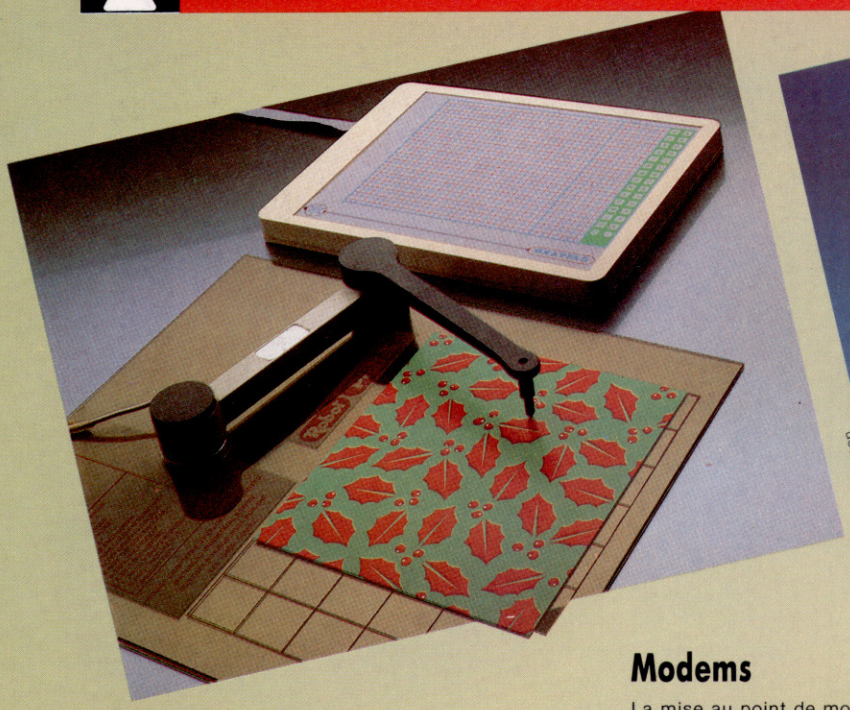
L'acheteur de périphériques doit surtout se préoccuper de la compatibilité du matériel — tout périphérique acheté doit être à même de fonctionner avec tout autre dispositif qui pourrait être acquis ultérieurement. Un exemple classique de ce problème concerne le Sinclair Spectrum. De nombreux propriétaires de Spectrum qui se sont décidés pour l'Interface 1 et un Microdrive ou deux ont découvert que certains de leurs périphériques — et même certains de leurs logiciels — ne fonctionnent pas lorsque l'Interface 1 est installée.

Si la compatibilité entre les périphériques est maintenue, l'acquisition de dispositifs complémentaires peut accroître considérablement les possibilités d'un système. Des périphériques adéquats peuvent vous permettre de concevoir un système informatique correspondant à vos exigences particulières, et d'évoluer selon vos besoins.

Chris Stevens

Systèmes de stockage

Le dispositif de stockage le plus répandu dans les systèmes de micro-informatique est l'unité à cassette. Elle a l'avantage d'être facile à utiliser et relativement bon marché, mais les programmes sont longs à charger, et il est difficile de conserver une liste précise du contenu d'une bande. Les lecteurs de disquettes sont plus rapides et plus fiables, mais aussi plus coûteux. La plupart des ordinateurs domestiques ne peuvent utiliser qu'un seul type de lecteur de disquettes, et certains d'entre eux sont très lents.
La plupart des ordinateurs utilisent encore des disquettes 5 $\frac{1}{4}$ pouces, mais celles de 3 et de 3 $\frac{1}{2}$ pouces deviennent de plus en plus populaires.
Le lecteur Oric/Atmos utilise par exemple des disquettes de 3 pouces avec une capacité de 160 K par face. Le système Torch Disk Pack transforme par exemple le BBC en un nouvel ordinateur, avec un microprocesseur Z80 qui vient compléter le 6502, et avec une capacité supplémentaire de mémoire de 64 K. Le Torch comprend aussi quatre programmes de gestion.
Par ailleurs, le Sinclair Spectrum n'autorise pas la connexion de lecteur de disquettes standard, bien que des sociétés indépendantes aient produit des interfaces spéciales. Sinclair a mis au point le système Interface I Microdrive qui peut stocker environ 85 K de données sur une bande. La bande est placée entièrement sous le contrôle de l'ordinateur, et tout élément de données peut être localisé en moins de 10 secondes. Cela correspond à un compromis entre le rendement d'un lecteur de disquettes et celui d'une unité à cassette, à un coût considérablement inférieur à celui du lecteur de disquettes le moins cher. Cette interface apporte l'autre avantage de fournir l'interface RS232 (non standard) et permet d'établir un réseau (liaison allant jusqu'à 64 micros).
Le Wafadrive de Rotronics est un rival pour le système Interface I. Il utilise également une bande, mais inclut les interfaces RS232 et Centronics et un programme de traitement de texte dans son prix d'achat. Il existe trois formats de bandes : 16, 64 et 128 K; la plus petite capacité correspond au temps d'accès le plus rapide.



Anderson-Jacobson

Périphériques graphiques

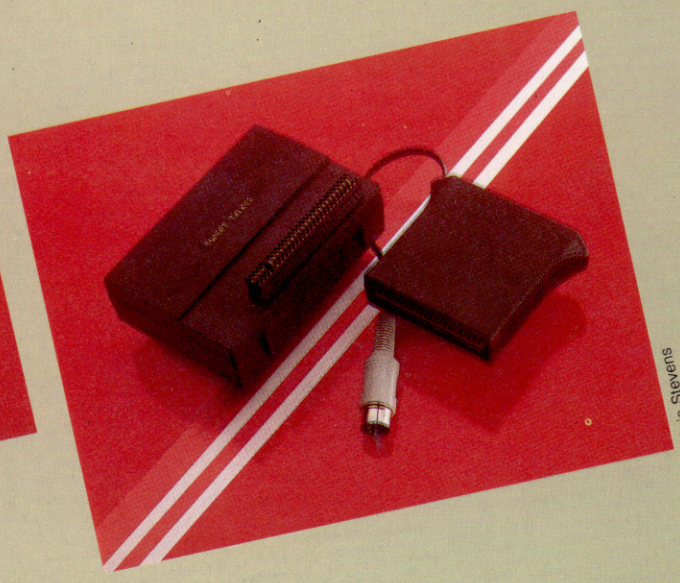
La réalisation de graphiques avec un ordinateur domestique est considérablement facilitée si l'on utilise l'un des nombreux périphériques graphiques existants. Les moins coûteux sont les crayons optiques, qui permettent de dessiner directement sur l'écran grâce à une cellule photoélectrique. Celle-là détecte la position du crayon optique lorsqu'il touche l'écran.
Le Stack Light Rifle use de la même technologie et peut remplacer un manche à balai dans certains jeux.
De nombreuses personnes ont des difficultés à dessiner sur un écran. Il est beaucoup plus facile de tracer des lignes sur une interface plate. Les tablettes graphiques utilisent un crayon spécial qui transmet à l'ordinateur tout mouvement tracé sur la tablette; cela signifie qu'elles peuvent servir à dessiner des images ou à calquer des images imprimées. Les traceurs numériques sont d'autres dispositifs de dessin; ils détectent la position d'un stylo placé au bout d'un bras mécanique grâce à la présence de résistances variables placées dans ce bras.
Les photos présentent la tablette graphique Grafpad, le traceur numérique Robot Plotter, le crayon optique Stack et le Light Rifle.

Modems

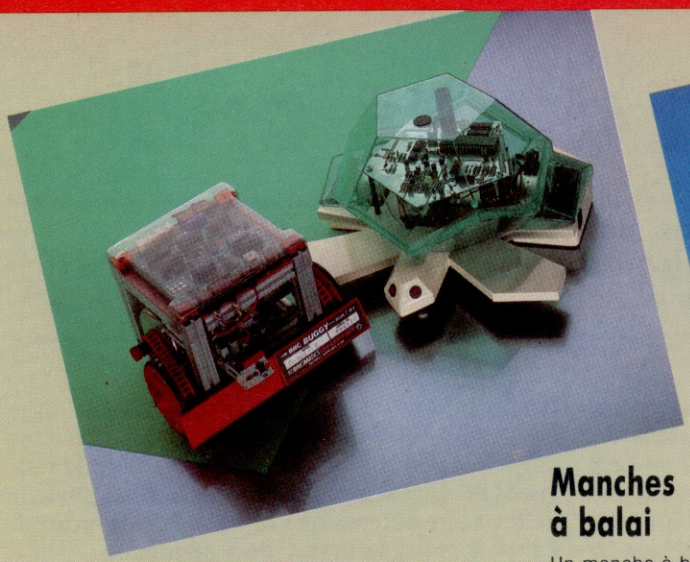
La mise au point de modems peu coûteux pour les ordinateurs domestiques permet aux utilisateurs de communiquer entre eux grâce au réseau téléphonique. Un grand nombre de modems ont été produits pour les machines équipées de l'interface RS232; avec le logiciel approprié, ces dispositifs permettent d'avoir accès à des bases de données publiques. Les modems peuvent aussi servir à communiquer avec d'autres utilisateurs via des bases de données dites « tableaux d'affichage » qui sont souvent mises au point à un niveau amateur par des mordus de micro-informatique. Cependant, le problème de compatibilité se pose ici encore — des vitesses différentes sont utilisées par différents tableaux d'affichage, et un modem qui peut utiliser une base de données publique est souvent inadéquat pour communiquer avec un tableau d'affichage.
Le Spectrum et le Commodore 64 n'ont pas d'interface RS232 intégrée et ne peuvent donc pas utiliser de modems standard. Commodore offre son propre modem pour le 64, et a mis au point son propre système, Compunet, pour créer un réseau reliant les utilisateurs de Commodore 64.
Les utilisateurs de modems doivent tenir compte du temps consacré à leurs communications, puisque les notes de téléphone peuvent devenir assez lourdes.

Synthétiseurs de parole

De nombreux ordinateurs domestiques peuvent « parler » : il suffit d'ajouter une unité de synthèse de la parole.
Les unités offertes peuvent être groupées en deux catégories : un premier type offre un vocabulaire fixe d'environ cent mots, tandis que l'autre type utilise des « allophones », ensemble de sons et de pauses différents à partir desquels les mots sont construits.
La gamme de synthétiseurs Currah de parole utilise le système allophone. Cette société produit des modules pour le Spectrum (Microspeech) et pour le Commodore 64 (Speech 64). Certains jeux Commodore et Spectrum, notamment ceux produits par Ultimate, ont un synthétiseur intégré — la parole se produit automatiquement si une unité Currah est connectée.



Chris Stevens



Commandés par ordinateur

Les ordinateurs peuvent facilement servir à commander des appareils. On cite souvent comme exemple la commande d'un système de chauffage central domestique. Cette commande est facile à mettre au point, mais la démarche n'est pas réellement justifiée puisqu'un interrupteur chronométré réussit à satisfaire presque tous les besoins. Les véhicules commandés par ordinateur sont beaucoup plus intéressants. La tortue Valiant est un véhicule roulant qui ressemble un peu à une tortue et qui peut produire les graphiques créés par le langage LOGO.
Ce dispositif peut accompagner le Spectrum ou le Commodore 64, et utilise un faisceau à infrarouges pour communiquer avec l'ordinateur.

Imprimante/traceur

Toute personne qui utilise un ordinateur pour la mise au point de programmes ou pour le traitement de texte comprendra rapidement que l'imprimante est un élément essentiel. La plupart des imprimantes sont soit matricielles, soit à marguerite. Les matricielles utilisent une grille de petits points pour construire chaque lettre. Elles permettent l'impression de graphiques. Elles sont rapides mais leur qualité d'impression est inférieure à celle des imprimantes à marguerite qui sont essentiellement des machines à écrire commandées par ordinateurs.
Il existe également un type d'imprimante/traceur commercialisé par Tandy, Atari ou Commodore. Ce dispositif, qui utilise un papier très étroit, est muni de quatre stylos à bille qui permettent de produire du texte ou des graphiques couleur. Les caractères sont dessinés de la même manière que les graphiques.
Mentionnons également l'imprimante thermique Epson P40. Cette imprimante est peu coûteuse, donne une qualité d'impression raisonnable et fonctionne avec des piles rechargeables. De nouveau, le papier est assez étroit, mais la P40 peut produire une impression sur 80 colonnes en mode compressé.

Manches à balai

Un manche à balai est généralement le premier périphérique acheté. La plupart des ordinateurs sont munis des interfaces adéquates, et des machines plus récentes offrent des manches à balai en version standard. La plupart des manches à balai utilisent des connecteurs à neuf broches de type D adoptés d'abord par les micros Commodore et Atari; ce type de connecteur a ensuite été retenu par de nombreuses sociétés indépendantes. Commodore a assez curieusement ignoré ses propres normes sur ses micros Plus/4 et 16, obligeant les acheteurs de ces machines à se procurer les nouveaux manches à balai Commodore.
Sinclair a récemment introduit l'Interface 2, une interface de manche à balai et un port de cartouche pour le Spectrum. Jusqu'à l'arrivée de ce dispositif, l'interface Kempston s'était imposée *de facto*. Malheureusement, les deux sont incompatibles, et de nombreux jeux populaires fonctionneront parfaitement avec l'interface Kempston mais refuseront de fonctionner avec l'Interface 2. Kempston a donc produit une interface compatible avec les programmes écrits pour les deux interfaces. Il est aussi possible d'acheter une interface programmable qui permette à l'utilisateur d'exécuter tout logiciel, qu'il ait été conçu ou non pour être utilisé avec un manche à balai. Dans ce cas, le manche à balai doit reproduire l'action du clavier. Cette interface est particulièrement indiquée pour tout propriétaire de Spectrum qui a une large collection de jeux.
Parmi les nombreux manches à balai offerts sur le marché, le plus curieux est probablement le nouveau RAT Cheetah. Celui-ci est relié à l'ordinateur au moyen d'un faisceau à infrarouges comme pour l'Exelvision. Le système Amstrad est lui aussi assez inhabituel. Le micro Amstrad ne possède qu'une seule prise de manche à balai; mais un second manche à balai peut être connecté au premier, permettant ainsi d'exécuter des jeux à deux joueurs.

Moniteurs

La plupart des ordinateurs domestiques sont utilisés — du moins initialement — avec un téléviseur comme écran d'affichage. Cela pose souvent des problèmes familiaux... De toute façon, la qualité d'affichage ainsi obtenue est souvent médiocre.
La solution consiste à utiliser un moniteur qui donne une meilleure qualité d'image.
L'utilisateur doit veiller à acheter le bon moniteur, puisqu'il existe deux normes principales : RVB et vidéo composite.
Les moniteurs vidéo composite sont utilisés avec les micros Atari et Commodore, tandis que d'autres marques adoptent le format RVB, qui donne une meilleure qualité d'image. Certains micros utilisent le haut-parleur du téléviseur pour produire des sons; ils auront donc besoin d'un moniteur muni d'un haut-parleur intégré. Plusieurs fabricants de téléviseurs produisent des appareils munis d'interfaces pour moniteurs.

Définition d'une onde

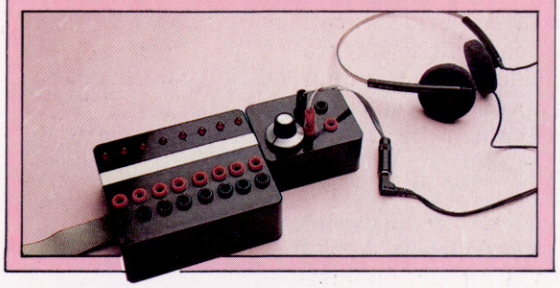
Il est possible de produire des signaux sonores à partir d'un convertisseur numérique/analogique. Nous allons étudier la production de divers signaux et déterminer la durée d'une note.

Nous pouvons tester le système à l'aide d'un court programme BASIC. En gros, le son est produit électriquement en appliquant une tension oscillante à un haut-parleur. Nous pouvons générer une sortie de tension oscillante à partir de notre convertisseur numérique/analogique en changeant le contenu du registre de données du port utilisateur de 0 à 255. Tapez le programme suivant et exécutez-le. Tournez le potentiomètre dans le sens des aiguilles d'une montre jusqu'à ce qu'un son puisse être entendu.

Le projet

Vous pouvez contrôler la sortie à l'aide d'un casque ou d'une chaîne hi-fi, en utilisant les prises situées près du potentiomètre sur le boîtier N/A. Pour le casque, utilisez la fiche stéréo. Procurez-vous une prise adéquate et soudez les deux pattes positives de cette prise à un conducteur connecté à la prise rouge du boîtier N/A. Pour une chaîne hi-fi : consultez le manuel de votre amplificateur pour localiser les connexions audio-IN et de mise à la terre et établissez la connexion. Suivez maintenant ces directives : connectez le boîtier tampon et le convertisseur N/A et branchez le boîtier tampon dans le port utilisateur; branchez les conducteurs du casque ou de la chaîne hi-fi dans les prises N/A; tournez le potentiomètre du convertisseur N/A complètement vers la gauche et appliquez la puissance du transformateur au boîtier.

Ian McKinnell



```
10 REM **** GENERATEUR SONORE BASIC CBM ****
20 DDR=56579:DATREG=56577
30 POKEDDR,255
35 N=1
40 POKE DATREG,0:FORI=1TON:NEXT:POKE
   DATREG,255:GOTO40

10 REM **** GENERATEUR SONORE BASIC BBC ****
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=255
35 N=1
40 ? DATREG=0:FOR I=1TON:NEXT:? DATREG=255:GOTO40
```

Notez que ce programme BASIC est doté d'une structure répétitive réunie sur une seule ligne pour produire une vitesse maximale. Une boucle de temporisation est insérée entre le moment où le registre de données est mis à 255 et celui où il est mis à 0. La valeur N de la ligne 35 définit la longueur de cette temporisation. Modifiez la valeur de N : vous noterez que la note du son produit baisse lorsque la valeur de N augmente.

La note la plus élevée qu'on peut obtenir à partir de ce programme sera produite lorsque la boucle de temporisation sera entièrement supprimée.

Si vous avez attribué différentes valeurs à N dans un programme donné, vous avez remarqué que faire varier de 1 la valeur N a un effet significatif sur la note du son produit. BASIC n'est pas assez rapide pour commander la vitesse d'oscillation de façon précise. Nous devons plutôt avoir recours au langage machine.

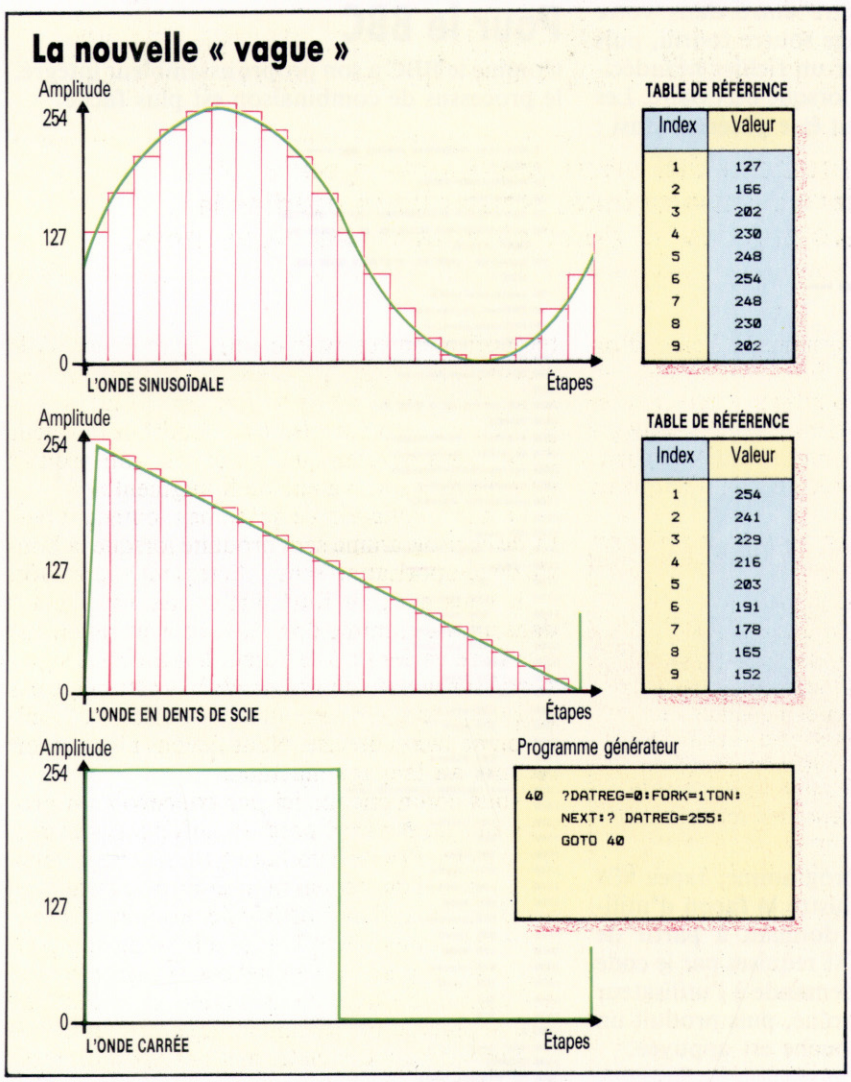
Nous commençons ici par concevoir un programme qui servira à produire différents signaux. La forme d'onde produite par le programme BASIC utilisé précédemment était une onde carrée. Il est cependant possible de produire d'autres formes qui modifient la « qualité » du son produit. Nous pouvons synthétiser numériquement des formes d'ondes sinusoïdales et en dents de scie en prenant un certain nombre d'échantillons de formes d'onde et en les mettant dans une table de référence. Le programme en code machine nécessaire pour placer ces échantillons à tour de rôle dans le registre de données est très simple.

Dans la production de sons, la synchronisation est cruciale. A la suite de chaque instruction apparaît le nombre de cycles machine nécessaires pour exécuter cette instruction. A partir de cette formule, nous pouvons calculer le nombre total de cycles machine nécessaires pour produire un cycle complet d'onde : nombre de cycles machine = 2 + (4 + 4 + 2 + 2 + 3) × étapes − 1 = 1 + 15 × étapes. Si l'onde est divisée en 80 étapes, le nombre de cycles machine requis pour produire un cycle d'onde est 1201.

Comme un cycle machine du 6502 est exécuté en un millionième de seconde environ, le nombre total de cycles d'onde pouvant être produits dans une seconde (c'est-à-dire la fréquence de la note) est donné par ce calcul : fréquence = 1 000 000/1201 = 832 Hz. Puisque le *do* de référence a une fréquence de 512 Hz, la note produite doit se situer quelques notes plus haut que le *do* de référence.

On peut constater que le nombre d'étapes d'échantillonnage dans lequel nous décidons de diviser notre onde a un effet direct sur la fréquence de la note finale. Doubler le nombre d'étapes d'échantillonnage diminuerait de moi-

```
        LDX#0          2
LOOP1
        LDA TABLE,X    4
        STA PORT       4
        INX            2
        CMP #STEPS     2
        BNE LOOP1      3
        (2 si boucle manque)
```



Formes d'ondes
L'onde sinusoïdale et l'onde en dents de scie sont créées en déterminant d'abord le nombre d'éléments (étapes) constituant un cycle. Ces éléments de l'amplitude de l'onde sont ensuite calculés et stockés dans une table de référence. Les valeurs peuvent alors être copiées de façon séquentielle dans le registre de données du port utilisateur et transmises au convertisseur N/A où elles deviennent des niveaux de tension. L'avantage de la table de référence est qu'elle permet d'effectuer d'avance de longs calculs; la formation de l'onde dure quelque temps; cela rend possible la production d'une gamme de fréquences couvrant plusieurs octaves. Sans la table de référence, la gamme des fréquences aurait été limitée à deux octaves.
(Cl. Liz Dixon.)

tié la fréquence de la note finale. Évidemment, plus le nombre d'étapes d'échantillonnage sera grand, meilleure sera la qualité du son que nous synthétiserons; mais nous devons toujours tenir compte de la fréquence finale maximale obtenue pour un nombre donné d'étapes.

Il est peu probable qu'un cycle d'onde soit assez long pour être audible; nous devons donc inclure un code qui servira à répéter la section de génération de forme d'onde un certain nombre de fois. Le nombre de répétitions peut être déterminé en définissant une valeur de compteur et en la décrémentant jusqu'à zéro. Pour donner un large ensemble de valeurs, un nombre à 16 bits stocké dans deux adresses adjacentes a été utilisé. En plus de ce code, les interruptions sont interdites au début du programme par SEI et autorisées de nouveau par CLI à la fin. Si les interruptions survenaient pendant l'exécution, cela rendrait imprécise la synchronisation du programme.

Cependant, nous pouvons interdire uniquement certaines interruptions; des interruptions non masquables, si elles surviennent pendant l'exécution du programme, peuvent toujours être la cause — malheureusement — de certaines erreurs de synchronisation.

Les données de forme d'onde doivent être placées en mémoire sous la forme d'une table de référence, chaque type de forme d'onde occupant quatre-vingts adresses consécutives. Dans la version du Commodore, l'onde sinusoïdale est située en mémoire à partir de l'adresse $C000; la table de dents de scie est située à $C050 et la table d'onde carrée commence en $C0A0. Le programme est conçu pour charger par défaut les données de la table d'onde sinusoïdale à l'aide d'un adressage indexé, mais nous pouvons passer à une autre table en modifiant le programme directement avec une instruction POKE à partir de BASIC. La partie LDA de LDA SINE,X est à l'adresse $C103. L'adresse de départ de la table de données à charger a son octet lo à l'adresse $C104 et son octet hi à l'adresse $C105. Pour modifier l'adresse de départ des données à charger, nous n'avons qu'à changer le nombre contenu dans $C104. Normalement, pour une onde sinusoïdale, cette adresse contiendrait 0; si nous désirons une onde en dents de scie, nous n'avons qu'à changer le contenu de $C104 en 80. Placer 160 dans cette adresse changerait l'onde en une onde carrée.

La version BBC est aussi conçue de façon que les tables de référence commencent au début d'une nouvelle page de mémoire. Puisque les tables commencent sur une nouvelle page, l'octet hi de l'adresse de départ de table est le même, et nous ne devons modifier que l'octet lo.

Pour le Commodore

```
                ;++++++++++++++++++++
                ;++++++++++++++++++++
                ;++                ++
                ;++  GENERATEUR    ++
                ;++    SONORE      ++
                ;++    CBM 64      ++
                ;++                ++
                ;++++++++++++++++++++
                ;++++++++++++++++++++
                ;
                PORT   = 56577      ;ADRESSE DE REGISTRE DE DONNEES
                STEPS  = 80         ;NOMBRE DE PAS
                *=$C000
                ;
                ;++++ ZONE DE TABLE DE DONNEES ++++
                SINE     ***=STEPS
                SAW      ***=STEPS
                SQUARE   ***=STEPS
                NUMBER   ***=2
                COUNT    ***=2
                ;
                ;++++ PROGRAMME PRINCIPAL ++++
                ;
78              SEI
AD F0 C0        LDA NOMBRE
8D F2 C0        STA COMPTE          ;DEF VALEUR DU COMPTE


AD F1 C0        LDA NOMBRE+1
8D F3 C0        STA COMPTE+1
                ;
                LOOP2
A2 00           LDX #$00
                LOOP1
BD 00 C0        LDA SINE,X          ;LIRE DONNEES
8D 01 DD        STA PORT            ;VERS PORT UTILISATEUR
E8              INX
E0 50           CPX #STEPS
D0 F5           BNE LOOP1           ;FIN D'UN CYCLE
                ;
                ;++++ DECREMENTATION ++++
AD F2 C0        LDA COUNT
38              SEC
E9 01           SBC #$01
8D F2 C0        STA COUNT
AD F3 C0        LDA COUNT+1
E9 00           SBC #$00
8D F3 C0        STA COUNT+1
D0 E0           BNE LOOP2           ;SI OCTET HI>0
A9 00           LDA #$00
CD F2 C0        CMP COUNT
D0 D9           BNE LOOP2           ;SI OCTET LO>0
58              CLI
60              RTS
```

Le code machine peut être entré dans votre machine en tapant le listage source fourni, puis en l'assemblant pour créer un fichier hexadécimal qui puisse être chargé lorsque nécessaire. Les tables de référence peuvent être générées ainsi :

```
900 REM **** PROGRAMME SONORE CBM ****
910 :
915 DN=8:REM POUR CASSETTE DN=1
920 IFA=0 THENA=1:LOAD"5SOUND.HEX",DN,1
999 :
1000 REM **** DEFINITION DES VALEURS DE DONNEES ****
1005 S=80          :REM NOMBRE D'ETAGES
1007 TB=12*4096  :REM DEBUT ZONE DONNEES
1008 :
1010 REM ** ONDE SINUSOIDALE **
1020 FOR I=0 TO S-1
1030 Y=127*SIN(X)+127
1040 POKE TB+I,Y
1045 X=X+2/S
1050 NEXT I
1060 :
1065 REM **** ONDE DENTS DE SCIE ****
1070 Y=255:TB=TB+S
1080 FOR I=0 TO S-1
1090 POKE TB+I,Y
1100 Y=Y-255/S
1110 NEXT I
1120 :
1125 REM **** ONDE CARREE ****
1130 Y=255:TB=TB+S
1140 FOR I=0 TO S/2-1
1150 POKE TB+I,Y
1160 NEXT I
1165 Y=0
1170 FOR I=S/2 TO S-1
1180 POKE TB+I,Y
1190 NEXT I
1999 :
2000 REM **** AFFICHAGE TABLES DONNEES ****
2005 TB=12*4096
2010 FORI=TB TO TB+3*S-1
2020 PRINTI,I-TB ,PEEK(I)
2030 NEXT
```

Après l'exécution de ce programme, tapez NEW puis ce programme qui illustre la façon d'utiliser le code machine, en donnant à partir de BASIC les adresses SYS et POKE requises par le code machine. Ce programme demande à l'utilisateur d'entrer le type d'onde voulue, puis produit un son chaque fois qu'une touche est appuyée.

```
10 REM **** SON CBM 64 ****
20 REM **** PROGRAMME EXEMPLE ****
30 :
40 DDR=56579:POKE DDR,255: REM TOUTES SORTIES
65 CL=49392 :REM COMPTEUR ADRESSE OCTET LO
67 TL=49412 :REM TAPER ADRESSE OCTET LO
70 SOUND=49396:REM ADRESSE DEPART DU PROGRAMME
75 REM ** VALEUR DU COMPTEUR **
80 NUM=80:NHI=INT(NUM/256):NLO=NUM-256*NHI
82 POKE CL,NLO:POKE CL+1:NHI
83 :
85 PRINTCHR$(147):REM EFFACER ECRAN
86 INPUT"WAVE TYPE (0)SINE (1)SAW (2)SQUARE";WT
87 POKE TL,WT+S
88 PRINT:PRINT"APPUYEZ UNE TOUCHE (RUN/STOP
   POUR SORTIR)"
90 GETA$:IFA$=""THEN90 :ATTENDRE TOUCHE
100 SYS SOUND:REM APPEL CODE MACHINE
110 IF A$="X" THEN 85
120 GOTO 90
```

Si vous n'avez pas d'assembleur ou si vous ne comprenez pas le langage assembleur, vous pouvez toujours utiliser le code machine, en tapant ce programme de chargement BASIC, et l'exécuter. Dans ce cas, vous pouvez omettre la ligne 920 du programme qui définit la table de référence.

```
10 REM **** CHARGEUR BASIC POUR SON ****
20 REM ****       CODE MACHINE        ****
30 FOR I=49396 TO 49449
40 READ A:POKE I,A
50 CC=CC+A
60 NEXT I
70 READ CS:IF CC<>CS THEN PRINT"ERREUR
   DE CONTROLE":END
100 DATA120,173,240,192,141,242,192
110 DATA173,241,192,141,243,192,162,0
120 DATA189,0,192,141,1,221,232,224,80
130 DATA208,245,173,242,192,56,233,1
140 DATA141,242,192,173,243,192,233,0
150 DATA141,243,192,208,224,169,0,205
160 DATA242,192,208,217,88,96
170 DATA9115:REM*TOTAL DE CONTROLE*
```

Pour le BBC

Comme le BBC a son propre assembleur intégré, le processus de combinaison est plus facile.

```
  5 REM **** PROGRAMME SONORE BBC ****
  8 MODE 7
 10 HIMEM=HIMEM-&0301
 20 MCX=HIMEM+1
 30 DDR=&FE62:? DDR=255:REM TOUTES SORTIES
 40 port=&FE60:REM REGISTRE DE DONNEES DU
    PORT UTILIS.
 50 steps=80  :REM NO. D'ETAPES DANS UNE
    ONDE
 60 table_start=MCX
 70 PROCset_up_tables
 80 PROCcode_machine
 90 PROCprogramme_exemple
999 END
1000 DEF PROCcode_machine
1010 :      -
1020 FOR optX=1 TO 3 STEP 3
1030 P%=MCX
1060 sin=P%:   P%=P%+steps
1070 scie=P%:  P%=P%+steps
1080 carree=P%:P%=P%+steps
1090 nombre=P%:P%=P%+2
1100 compte=P%: P%=P%+2
1110 [
1120 OPT optX
1130 / **** DEBUT PROGRAMME PRICIPAL ****
1150 .sound
1160       SEI
1170       LDA nombre
1180       STA compte
1190       LDA nombre+1
1200       STA compte+1
1220  .loop2
1230       LDX #&00
1240 .loop1
1250       LDA sin,X
1260       STA port
1270       INX
1280       CPX #steps
1290       BNE loop1
1310 / **** ABAISSE COMPTEUR ****
1320
1330       LDA compte
1340       SEC
1350       SBC #&01
1360       STA compte
1370       LDA compte+1
1380       SBC #&00
1390       STA compte+1
1400       BNE loop2
1410       LDA #&00
1420       CMP compte
1430       BNE loop2
1440       CLI
1450       RTS
1455 ]
1460 NEXT optX
1480 ENDPROC
2000 DEF PROCdéf_table
2020 REM **** ONDE SINUS ****
2025 x=0
2030 FOR I=0 TO steps-1
2040 y=127*SIN(x)+127
2050 ?(table_start+I)=y
2060 x=x+2*PI/steps
2070 NEXT I
2090 REM **** ONDE DENTS DE SCIE ****
2100 y=255:table_start=table_start+steps
2110 FOR I=0 TO steps-1
2120 ?(table_start+I)=y
2130 y=y-255/steps
2140 NEXT I
2160 REM **** ONDE CARREE ****
2170 y=255:table_start=table_start+steps
2180 FOR I=0 TO steps/2-1
2190 ?(table_start+I)=y
2200 NEXT I
2220 y=0
2230 FOR I=steps/2 TO steps-1
2240 ?(table_start+I)=y
2250 NEXT I
2270 REM **** AFFICHER TABLE DONNEES ****
2280 table_start=MCX
2290 FOR I=table_start TO table_start+3*steps-1
2300 PRINT~I,~(I-table start),? I
2310 NEXT I
2330 ENDPROC
3000 DEF PROCprogramme_exemple
3020 counter=MCX+3*steps: ADRESSE OCTET
     LO DU COMPTEUR
3030 type=loop1+1: REM TAPER ADRESSE
     OCTET LO
3040 compte_value=80
3050 compte_hi=compte_value DIV 256
3060 compte_lo=compte value_MOD 256
3070 ?compteur=compte_lo
3080 compteur?1=compte_hi
3090 CLS
3100 INPUT"TYPE D'ONDE (0) SINUS (1) DENTS DE SCIE (2)
     CARREE";wave
3100 ?type=wave*steps
3110 ?type=onde*steps
3120 REPEAT
3125 PRINT "APPUIE SUR UNE TOUCHE (X POUR SORTIR)"
3130 A$=GET$
3140 CALL son
3150 UNTIL A$="X"
3160 GOTO 3090
```

Dites-le avec des chiffres

LOGO n'est pas un langage de premier choix pour les applications supposant beaucoup de calculs, mais il offre un tableau impressionnant de primitives de calcul numérique.

Presque toutes les applications LOGO permettent le calcul en valeurs entières et réelles (décimales) à l'aide des opérateurs intercalés +, -, *, /. Ces derniers doivent leur nom au fait qu'ils sont entre les chiffres concernés, par exemple : 3+4. Certaines versions LOGO comprennent également des opérations arithmétiques en préfixe telles que SOMME 3 4. Un des avantages de cette notation est d'être cohérente avec les autres commandes LOGO.

LOGO MIT ne comprend que des opérations arithmétiques intercalées, mais il est simple de programmer des opérateurs en préfixe. Définissez SOMME et PRODUIT et essayez-les :

```
POUR SOMME :A :B
   RÉSULTAT :A + :B
FIN

POUR PRODUIT :A :B
   RÉSULTAT :A * :B
FIN
```

L'ordre de priorité entre opérateurs suit les règles mathématiques habituelles. Tout ce qui figure entre parenthèses est effectué en premier, et les quatre opérations classiques suivent dans l'ordre de priorité décroissant :

```
AFFICHE (3 + 4) * 5
AFFICHE 3 + 4 * 5
```

Essayez maintenant les syntaxes préfixées :

```
AFFICHE PRODUIT 5 SOMME 3 4
AFFICHE SOMME 3 PRODUIT 4 5
```

Cela met en évidence un autre avantage des préfixes : ils n'ont pas besoin de règles de priorité et la ligne est analysée de la même manière que toute autre ligne de commande LOGO.

L'opération classique de division (/), donne en résultat un nombre réel. Le QUOTIENT et le RESTE, sont également utiles pour les calculs avec des nombres entiers.

```
QUOTIENT 47 5 donne 9
RESTE 47 5 donne 2
```

Une méthode classique de conversion d'un nombre en base 10 en un nombre binaire est de le diviser indéfiniment par 2, jusqu'à zéro. Par exemple, pour convertir 12 en binaire :

```
12/2 = 6; reste = 0
 6/2 = 3; reste = 0
 3/2 = 1; reste = 1
 1/2 = 0; reste = 1
```

En lisant les restes des opérations depuis la fin vers le début, nous trouvons que l'équivalent binaire de 12 est 1100.

Nous pouvons adapter cette technique sur LOGO par QUOTIENT et RESTE. En plaçant l'instruction AFFICHER *après* l'appel récursif, nous obtenons les restes des opérations dans l'ordre voulu.

```
POUR BINAIRE :X
   SI :X = 0 ALORS STOP
   BINAIRE QUOTIENT :X 2
   AFFICHE 1 RESTE :X 2
FIN
```

Il existe deux opérations pour arrondir un nombre : ENTIER et ARRONDI. ENTIER donne la partie entière d'un nombre, ignorant ce qui vient après la virgule. ARRONDI fait une approximation sur la valeur du nombre, vers l'entier supérieur ou inférieur.

Les procédures suivantes calculent les intérêts composés d'un placement selon un certain taux d'intérêt. Dans MAGOT.AFFICHE, ENTIER est utilisé pour obtenir les francs, et ARRONDI, pour effectuer une approximation des centimes.

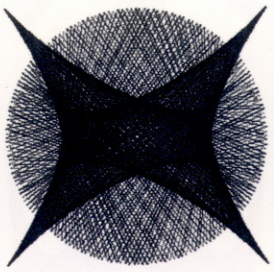
```
POUR INTÉRÊTS.COMPOSÉS :PRINCIPAL :TAUX :ANNÉES
   SI :ANNÉES = 0 ALORS MAGOT.AFFICHE
   :PRINCIPAL STOP
   INTÉRÊTS.COMPOSÉS :PRINCIPAL * (1 + :TAUX / 100)
      TAUX :ANNÉES - 1
FIN

POUR MAGOT.AFFICHE /ARGENT
   FAIT «FRANCS ENTIERS :ARGENT
   FAIT «CENTIMES ARRONDIR ( :ARGENT-
      :FRANCS) * 100
   (AFFICHE :FRANCS «FRANCS :CENTIMES
      «CENTIMES)
FIN
```

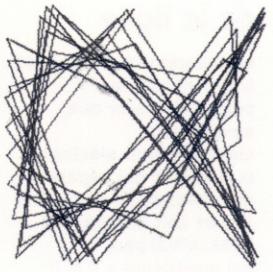
Tester le temps

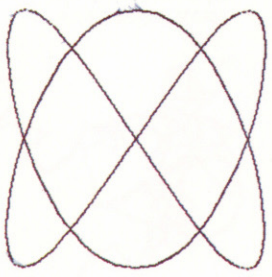
Nous avons déjà utilisé =, <, et > comme tests logiques de certaines procédures. Les opérations logiques LES.DEUX, L'UNE et NON- sont utilisables pour combiner d'autres tests. La condition logique LES.DEUX est vraie lorsque ses deux entrées le sont. L'UNE est vraie lorsque l'une des entrées l'est ; NON- est vraie lorsque son entrée ne l'est pas. Nous obtenons :

```
SI L'UNE :X > 0 :X = 0 ALORS AFFICHE «POSITIF
SI NON- :X < 0 ALORS AFFICHE «POSITIF
SI LES.DEUX :X > 0 :X < 100 ALORS AFFICHE
   [ENTRE 0 ET 100]
```

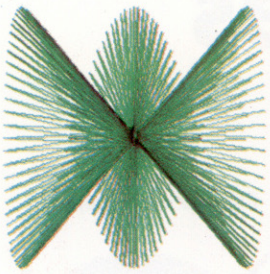
Figures de Lissajous

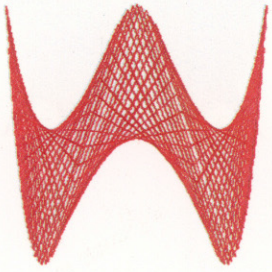






Figures de Lissajous





L'opération NOMBRE? donne VRAI lorsque son entrée est un nombre, et FAUX dans le cas contraire. Nous l'utilisons dans la procédure NOMBRE.PREMIER? qui donne comme résultat VRAI si le nombre entré est bien un nombre premier, et FAUX dans le cas contraire. Elle commence par vérifier que la saisie est bien un nombre, et que ce nombre est supérieur à 2. TEST.PREMIER vérifie qu'une valeur entière comprise entre la racine carrée du nombre et 2 pourra être divisée et donner le nombre exactement.

```
POUR PREMIER? :NON
  SI NON- NOMBRE? :NON ALORS AFFICHER [CE N'EST PAS
    NOMBRE] STOP
  SI :NON < 2 ALORS RÉSULTAT «FAUX
  RÉSULTAT TEST.PREMIER :RÉSULTAT
  RACINE CARRÉE NON ENTIER :NON
FIN

«POUR TEST.PREMIER :NON :FAIT
  SI :FAIT = 1 ALORS RÉSULTAT «VRAI
  SI (RESTE :NON :FAIT) = 0 ALORS RÉSULTAT « FAUX
  RÉSULTAT TEST.PREMIER :NON :FAIT - 1
FIN
```

Un pas au-delà de la ligne

Le théorème du « matelot ivre » dit que la probabilité pour que, après *n* pas complètement aléatoires, la distance qui sépare le marcheur de son point de départ soit inférieure à RACINE.CARRÉ(N) pas, est supérieure à 0,5. Il s'agit d'une estimation statistique fondée sur un très grand nombre de pas. Logo vous permet de le vérifier par vous-même :

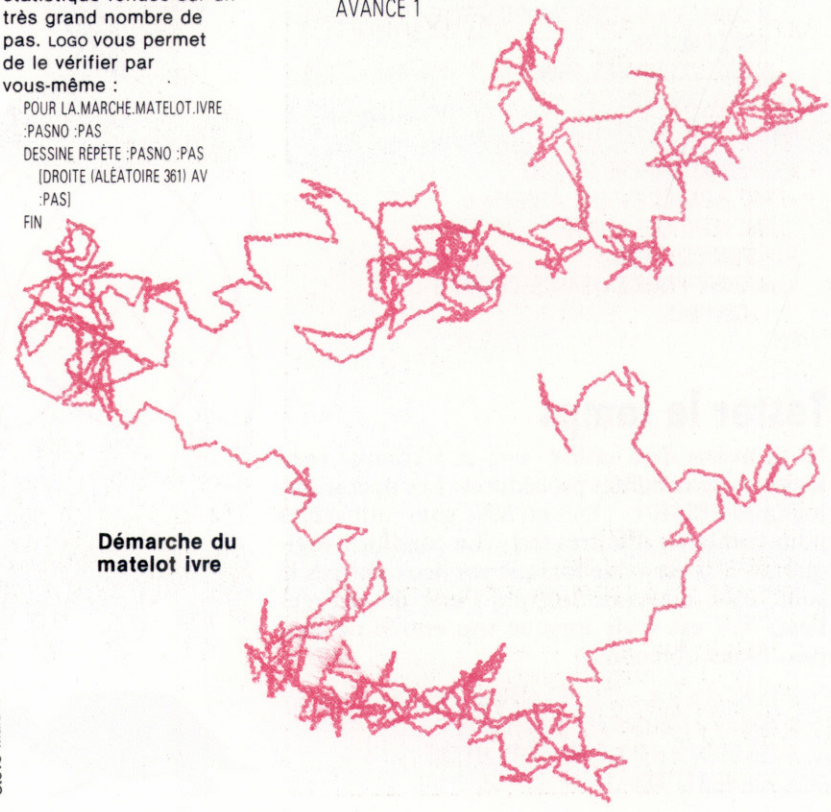
```
POUR LA.MARCHE.MATELOT.IVRE
:PASNO :PAS
DESSINE RÉPÈTE :PASNO :PAS
  [DROITE (ALÉATOIRE 361) AV
  :PAS]
FIN
```

Nombres aléatoires

ALÉATOIRE n produit un nombre aléatoire compris entre 0 et $n-1$. La procédure IVRE.MORT fait tituber la tortue sur l'écran et elle tourne à chaque pas selon un angle aléatoire.

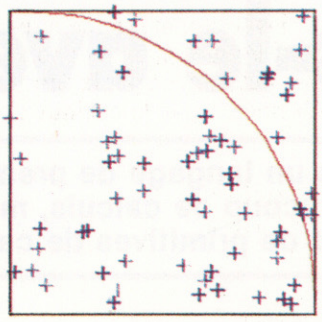
La valeur en entrée, A, donne l'angle maximal possible. Si vous exécutez cette procédure, vous verrez que la tortue effectue des sortes de cercles, se déplaçant sur la gauche ou sur la droite selon la valeur assignée à A.

```
POUR IVRE.MORT :A
  AVANCE 1
  DROITE (- :A/2 + ALÉATOIRE :A)
  IVRE.MORT :A
FIN
```



Démarche du matelot ivre

Steve Malone



Le nombre π par Monte-Carlo

La méthode dite « de Monte-Carlo » est une technique de résolution de problèmes mathématiques basée sur l'utilisation de nombres aléatoires.

Nous allons vous en donner un exemple en cherchant une approximation pour la valeur de π. Notre illustration montre un quart de cercle tracé dans un carré. La surface de ce carré est de 100 * 100 unités de carré. La surface du quart de cercle est $(1 \div 4) * \pi * 100 * 100$ unités de carré. Le rapport des surfaces cercle/carré est égal à $\pi \div 4$. Laissez tomber une aiguille au hasard dans le cercle cent fois de suite, et comptez combien de fois l'aiguille tombe dans le quart de cercle. Appelez cette valeur DEDANS. La valeur DEDANS/1000 devrait être approximativement la même, celle du résultat de : cercle ÷ carré, c'est-à-dire $\pi \div 4$.

```
POUR MC
  DESSINE
  LÈVEPLUME
  FAIT «DEDANS 0
  MC1 1000 100 100
  (AFFICHE [LA VALEUR DE PI EST] 0,004 * ( :DEDANS))
FIN
POUR MC1 :NON :XNO :YNO
  SI /NO = 0 ALORS STOP
  POINT.ALÉATOIRE :XNO :YNO
  SI DEDANS? ALORS FAIT «DEDANS :DEDANS + 1
  MC1 :NO - 1 :XNO :YNO
FIN
```

La procédure MC se contente de mettre en place les expressions conditionnelles, d'appeler MC1 et d'afficher les résultats. MC1 fait le plus gros du travail, appelle POINT.ALÉATOIRE pour positionner la tortue et incrémente DEDANS lorsque le point est bien dans le cercle. Cela continue jusqu'à ce que la procédure ait été effectuée le nombre voulu de fois.

```
POUR POINT.ALÉATOIRE :XNO :YNO
  POSITIONNERXY ALÉATOIRE :XNO ALÉATOIRE :YNO
FIN
POUR DEDANS?
  SI (COORDX * COORDX + COORDY * COORDY) < 10000
  ALORS RÉSULTAT «VRAI
  RÉSULTAT «FAUX
FIN
```

POINT.ALÉATOIRE positionne la tortue de manière aléatoire dans le carré, et DEDANS? vérifie si la tortue est dans le cercle. Ces procédures prendront du temps, mais vous trouverez en fin de compte une valeur pour π égale à 3,15999.

Les courbes de Lissajous sont une famille intéressante de courbes dans lesquelles la coordonnée x de chaque point est déterminée par la fonction sinus, et la coordonnée y par le cosinus :

```
POUR LJ: COEFF1 :COEFF2 :PAS
   DESSINE LÈVEPLUME CACHE.TORTUE
   POS :COEFF1 :COEFF2 0 POSEPLUME
   LJ1 :COEFF1 :COEFF2 0 :PAS
FIN

POUR POS :COEFF1 :COEFF2 :ANGLE
   FAIT «X 100 * SIN ( :COEFF1 * :ANGLE)
   FAIT «Y 100 * COS ( :COEFF2 * :ANGLE)
   POSITIONNEXY :X :Y
FIN

POUR LJ1 :COEFF1 :COEFF2 :ANGLE :PAS
   POS :COEFF1 :COEFF2 :ANGLE
   LJ1 :COEFF1 :COEFF2 ( :ANGLE + :PAS) :PAS
FIN
```

Variantes de Logo

Les versions LCSI sont en arithmétique préfixée. LOGO Atari comporte SOMME et PRODUIT. LOGO Spectrum comprend également DIV; et LOGO Apple, QUOTIENT (ces deux derniers opérateurs correspondant à QUOTIENT de LOGO MIT).

INT est utilisé pour ENTIER. NOMBREP est utilisé pour NOMBRE? Les opérateurs logiques ont les noms plus courants de ET, OU et NON.

SI a une syntaxe différente — SI :X = 0 AFFICHE «ZÉRO.
TAPE est utilisé à la place de AFFICHE1.
DONNEPOSITION (suivi d'une liste) est utilisé pour POSITIONNEXY.
Utilisez DESSINE au lieu de TRACE.

Exercices Logo

1. Écrire une procédure qui donne la puissance nième d'un nombre, de sorte que PUISSANCE 4 2 donne 16.
2. Écrire un ensemble de procédures pour convertir un nombre décimal en hexadécimal (utilisez une technique semblable à l'exemple binaire, mais divisez par 16).
3. Écrire une procédure PAIR? qui donnera VRAI si le nombre est pair, et FAUX si le nombre ne l'est pas.
4. Utilisez la méthode de Monte-Carlo pour trouver la surface délimitée par la courbe $y = x^2$ entre $x = 0$ et $x = 10$.

Réponses aux exercices

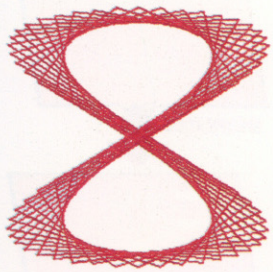
1. Modifiez le jeu pour donner le contrôle au clavier. Modifiez DONNE.DÉMONS, SURVEILLANCE, DÉTECTION. Supprimez DIRECTION.MANETTE. Ajoutez DÉPLACE et LIRE.FRAPPE.

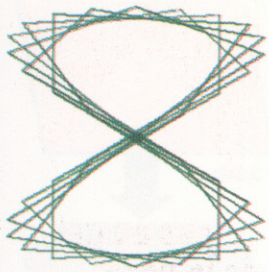
```
POUR DONNE.DÉMONS
LORSQUE AU.DELÀ :MOUTON1 :GRILLAGE
  [DONNE.VITESSE 0]
LORSQUE AU.DELÀ :MOUTON2 :GRILLAGE
  [DONNE.VITESSE 0]
LORSQUE TOUCHE :MOUTON1 :MOUTON2
  [DONNE.VITESSE 0]
LORSQUE TOUCHE :CHIEN :MOUTON1 [DONNE.VITESSE 0]
LORSQUE TOUCHE :CHIEN :MOUTON2 [DONNE.VITESSE 0]
   FIN

POUR SURVEILLANCE
   DÉPLACE LIRE.FRAPPE
   SI :VITESSE = 0 [DÉTECTION]
   SURVEILLANCE
FIN

POUR DÉTECTION
   SI COND AU.DELÀ :MOUTON1 :GRILLAGE [DEMANDE
     :MOUTON1 [AR 10 DR 90]]
   SI COND AU.DELÀ :MOUTON2 :GRILLAGE [DEMANDE
     :MOUTON2 [AR 10 DR 90]]
   SI COND TOUCHE :MOUTON1 :MOUTON2 [REBONDIR]
   SI COND TOUCHE :CHIEN :MOUTON1 [DEMANDE
     MOUTON1 [DR 90]]
   SI COND TOUCHE :CHIEN :MOUTON2 [DEMANDE
     :MOUTON2 [DR 90]] DONNE.VITESSES
FIN
POUR DÉPLACER :COM
   SI :COM = «W [DEMANDE :CHIEN [DONNEDIR 0]]
   SI :COM = «S [DEMANDE :CHIEN [DONNEDIR 90]]
   SI :COM = «Z [DEMANDE :CHIEN [DONNEDIR 180]]
   SI :COM = «A [DEMANDE :CHIEN [DONNEDIR 270]]
   SI :COM = «Q [DEMANDE :TORTUE [TRACE.CAGE]]
FIN
   POUR LIRE.FRAPPE
   SI FRAPPE [SAISIE RC]
   RÉSULTAT »
FIN
```

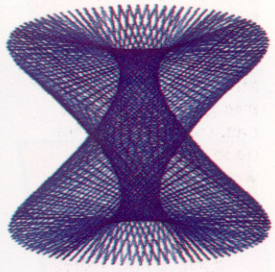
2. Le jeu du météorite : définissez la forme graphique 1 comme météorite et la forme 2 comme vaisseau spatial.

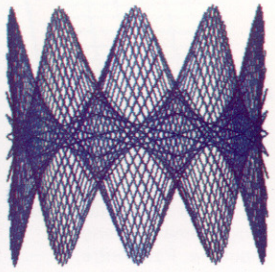
```
POUR JOUER
   DESSINE FS
   DONNE 0 1 [-100 80] 180 199
   DONNE 1 1 [-0 80] 180 199
   DONNE 2 1 [100 90] 180 199
   DONNE 3 2 [0-80] 90 50
   DONNE.DÉMONS
   DÉPLACEMENT.ALÉATOIRE 0
FIN
POUR POSITION :NO :FORME :POS :DIR :VAISSEAU
   DIRE :NO DONNE.FORME :FORME
   LÈVEPLUME DONNE.POSITION :POSITION
   DONNEDIR :DIR
   ST :DONNEVAISSEAU :VAISSEAU
FIN
POUR DONNE.DÉMONS
   LORSQUE TOUCHE 0 3 [BANG]
   LORSQUE TOUCHE 1 3 [BANG]
   LORSQUE TOUCHE 2 3 [BANG]
   LORSQUE 15 [DIRECTIONMANETTE]
FIN
POUR BANG
   DIRE [0 1 2 3]
   DONNEVAISSEAU 0 SS
   AFFICHE «AFFICHER»
   AFFICHE «ANÉANTISSEMENT
FIN
POUR DIRECTION.MANETTE
   SI (MANETTE 1) < 0 [STOP]
   DEMANDE 3 DONNE.DIRECTION 45 * MANETTE 1]
FIN
POUR DÉPLACEMENT.ALÉATOIRE :NO
   SI VITESSE = 0 [(AFFICHE «SCORE :NO) STOP]
   DEMANDE ALÉATOIRE 3 [DONNEDIR 145 + ALÉATOIRE 70]
   DÉPLACEMENT.ALÉATOIRE :NO + 1
FIN
```

Figures de Lissajous



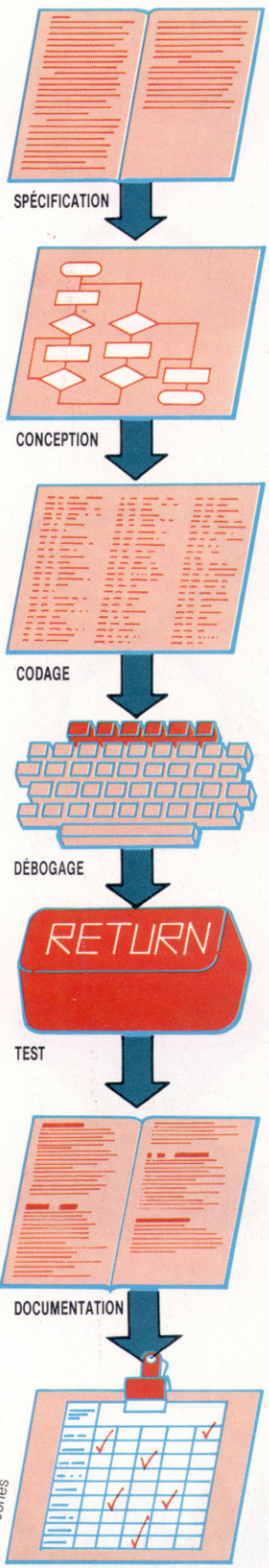






Concevoir l'ordre

Pour écrire des programmes plus longs et plus ambitieux, nous allons voir de nouvelles techniques qui nous permettront de structurer des programmes en langage d'assemblage.



Kevin Jones

Étapes de la conception
Il est difficile d'observer les règles de structure en programmation langage machine. Par contre, il n'est pas difficile de développer des programmes selon une bonne conception, et l'on y gagne en clarté et en temps de déboguage.

Il a beaucoup été question, dans ce cours, de conception de programmes, de construction modulaire et de programmation structurée dans le cadre des langages évolués. A ce niveau, il n'y a en général plus de structures de contrôle, telles que WHILE... WEND et IF... THEN... ELSE en BASIC, pour imposer au code un minimum de structure. Il n'y a pas non plus de notation pratique, et rien pour caractériser les données. Pis, un programme en langage d'assemblage atteint couramment, en termes de nombre d'instructions, une longueur supérieure de six à dix fois à celle d'un programme évolué. Surtout, il est bien plus facile de commettre des erreurs et il est possible d'effacer, par une erreur dans un seul octet, toutes les données contenues sur un disque. Pour rendre la programmation en langage machine moins décourageante, nous considérons ici une approche plus positive.

Il n'y a rien de particulièrement nouveau dans la technique de programmation structurée. La nouveauté et l'originalité apparente résident dans le fait que le monde de la micro-informatique, s'il est surtout constitué d'amateurs, devient actuellement de plus en plus professionnel, et donc plus sensible à la bonne qualité. Rien ne met mieux cela en évidence que le *débogage* d'un programme en langage machine assemblé à la main, non structuré et non documenté, que vous avez créé des mois auparavant et laissé de côté.

Les étapes de la conception d'un programme

• Spécification du problème. A ce stade, le programmeur en langage d'assemblage doit être particulièrement attentif à la spécification des entrées et sorties. Les périphériques sont souvent contrôlés directement — surtout le clavier et l'écran — de sorte qu'il faut considérer les signes qui sont utilisés. Il peut aussi y avoir des contraintes de synchronisation.

On ne dispose pas toujours de routines pratiques pour convertir les chaînes d'octets qui entrent ou sortent, dans une forme utilisable par le programme — par exemple, convertir une chaîne de caractères ASCII en un nombre décimal sous la forme binaire. Aussi est-il important de spécifier non seulement la forme dans laquelle les données se présentent, mais aussi celle qui est requise par le reste du programme.

• Conception du programme. Considérons à présent les processus qui transformeront une entrée spécifiée de programme en sortie spécifiée. Ceux-ci devront être regroupés, là où c'est possible, en modules indépendants, avec les données requises par chaque processus. Il existe deux méthodes principales pour « décomposer » un programme en modules : *de bas en haut,* où vous collectez un ensemble de modules qui vous paraissent utiles dans le contexte du programme, pour les assembler ensuite; *de haut en bas,* où le programme est successivement décomposé en unités de plus en plus petites, en se concentrant sur la fonction de chaque unité plutôt que sur la manière dont elle est réalisée, jusqu'à ce que le processus ne puisse plus être utilement poursuivi. C'est à ce point seulement que vous commencez à considérer la manière d'assembler chaque module en code.

La conception de bas en haut a le grand avantage d'utiliser des modules de bibliothèque, qui sont faciles à assembler. Le résultat a des chances d'être plus efficace en utilisation de mémoire. Les inconvénients viennent du programme dans son ensemble, qui sera souvent plus difficile à déboguer et à tester, et qui sera moins compréhensible. La conception de haut en bas conduit à des programmes mieux structurés, et chaque étape du processus peut être testée séparément à l'aide de petites routines qui prennent la place de modules non encore écrits, en acceptant simplement les entrées et en fournissant les sorties dans la forme correcte, sans effectuer aucun traitement. Mais les programmes tendent à utiliser plus de mémoires, et les routines développées n'ont guère de chance de trouver d'autres applications immédiatement.

A l'intérieur de chaque module, les données requises, leurs structures et les algorithmes doivent être spécifiés. Un organigramme est utile à ce niveau pour représenter des algorithmes, mais on trouve souvent plus facile de travailler dans une sorte de langage plus évolué, appelé pseudo-code. C'est généralement le PASCAL qui sert de base à ce pseudo-code. Mais il n'y a aucune raison de ne pas utiliser le BASIC. Cela nous permet de concevoir des algorithmes et des données d'une façon qui nous est familière, et confine les travaux de bas niveau à la tâche relativement simple de traduire l'algorithme du pseudo-code en langage d'assemblage. C'est bien plus facile que d'essayer de concevoir et de coder en langage d'assemblage en même temps.

• Codage. Si les routines ont été bien conçues, ce stade sera probablement le plus facile et le moins long de tous. Pour traduire un algorithme

de haut niveau en code de bas niveau, il est essentiel que les structures de contrôle utilisées au haut niveau soient maintenues au bas niveau, en évitant d'abuser de BRA et JMP. Rappelez-vous que le temps que vous gagnez en écrivant un code non structuré est à coup sûr perdu au stade d'un débogage frustrant. Dans le tableau ci-contre, nous donnons quelques exemples de la façon dont les structures de contrôle habituelles peuvent être codées — en supposant, pour simplifier, que les données sont sur 8 bits.

Le codage avec de telles structures de contrôle pose un problème : le programme est plus long. Si on n'est pas limité par la place, il n'y a pas lieu de l'économiser; les codes plus courts ne correspondent généralement pas à des temps d'exécution courts, mais impliquent un plus long développement et un débogage plus fastidieux. Lorsque l'espace est limité, il vaut mieux écrire un code plus long et structuré, et introduire une étape supplémentaire d'optimisation là où le code, qui fonctionne, peut être raccourci pour prendre en compte des circonstances particulières tout en maintenant autant que possible la structure essentielle.

• Débogage. A ce stade, chaque module est testé séparément pour s'assurer qu'il donne les sorties appropriées pour des entrées valides. Pour voir ce qui se passe, il est nécessaire d'inspecter le contenu des registres et des emplacements mémoire utilisés par le programme, et éventuellement de les changer. Il est presque impossible de déboger un programme d'assemblage sans l'aide d'utilitaires pour mettre et ôter des points d'interruptions. Ceux-ci permettent d'exécuter le programme jusqu'à la prochaine interruption, puis de vider les registres et d'inspecter et modifier leurs contenus.

• Test. Une fois que chaque module a été testé et débogué, tout le programme doit être assemblé et testé avec les données appropriées. C'est beaucoup plus facile lorsqu'on sait que toutes les parties fonctionnent correctement.

• Documentation. Les programmes en langage d'assemblage sont plus difficiles à comprendre que les programmes évolués; la documentation y est donc encore plus importante. En particulier, il est vital de documenter l'utilisation de la mémoire, celle des piles (surtout pour le passage des paramètres) ainsi que celle des registres dans les sous-programmes.

• Maintenance. Si un programme doit être utilisé pendant quelque temps, il nécessitera sans doute des révisions ponctuelles, soit pour supprimer des erreurs qui apparaîtront, soit pour l'améliorer. C'est à ce stade que le temps passé lors d'une conception soignée et en documentation se révélera réellement payant. Si le programme est mal conçu et/ou médiocrement documenté, il vaudra mieux le réécrire complètement plutôt que de tenter de le modifier.

Il nous faut à présent un projet pour appliquer ces techniques : pour notre première expérience de programmation structurée en langage d'assemblage, rien ne sera plus approprié qu'un moniteur/débogueur en langage machine. Si vous n'avez pas encore utilisé d'assembleur, vous vous familiariserez avec les avantages à attendre d'un moniteur/débogueur.

Il donne essentiellement au programmeur en langage machine une sorte de facilité d'écriture que le programmeur en BASIC croit naturelle — en fait, il s'agit de cette possibilité de surveiller et de changer les contenus de la mémoire.

Prochainement, nous utiliserons les différentes étapes de conception et de développement de notre projet pour créer une aide à la programmation importante et très utile.

Squelette basic
Il n'existe pas de structure de contrôle en langage d'assemblage, d'où l'intérêt de prendre des méthodes qui ont fait leurs preuves dans les langages évolués. Les structures présentées ici peuvent servir, à l'exclusion de toutes autres, aussi bien dans les langages évolués que ceux qui le sont beaucoup moins.

Structures de contrôle

Structure IF...THEN...ELSE

Pseudo-code	Langage d'assemblage
	THREE FCB 3
	.....
IF NUM1 = 3	IF LDA NUM1
THEN	CMPA THREE
routine1	BNE ELSE
	THEN .....
ELSE	* routine1
routine2	BRA ENDIF
ENDIF	ELSE .....
	* routine2
	ENDIF .....

Structure WHILE...WEND

Pseudo-code	Langage d'assemblage
WHILE NUM1 < =3	WHILE LDA NUM1
routine répétée	CMPA THREE
WEND	BGT WEND
	routine répétée
	BRA WHILE
	WEND .....

Structure REPEAT...UNTIL

Pseudo-code	Langage d'assemblage
REPEAT	REPEAT .....
routine répétée	* routine répétée
UNTIL NUM1 < 3	LDA NUM1
	CMPA THREE
	BGE REPEAT
	UNTIL .....

Structure FOR...NEXT

Pseudo-code	Langage d'assemblage
FOR NUM1=1 TO NUM2	LDA NUM2
routine répétée	FOR .....
NEXT NUM1	* routine répétée
	DECA
	BGT FOR
	NEXT .....

Carte du Z80 (suite)

Voici, avec l'aimable autorisation de Zilog Inc., une autre partie de la carte référence du programmeur Z80. A ne pas manquer pour tous ceux qui veulent programmer.

Groupes d'arithmétique générale et de contrôle d'UC

Arithmétique générale

Acc. ajust. décimal « DAA »	27
Acc. complément « CPL »	2F
Acc. négat. « NEG » (complément à 2)	ED 44
Drapeau de retenue compl. « CCF »	3F
Drapeau de retenue « SCF »	37

Contrôle UC

« NOP »	00	
« HALT »	76	
Invalide INT « (DI) »	F3	
Valide INT « (EI) »	FB	
Met INT en mode 0 « IM 0 »	ED 46	Mode 8080A
Met INT en mode 1 « IM 1 »	ED 56	Repart à l'emplacement 0038_H
Met INT en mode 2 « IM 2 »	ED 5E	Appel indirect utilisant registre à 1 et 8 bits à partir du dispositif d'interruption comme pointeur.

Mnémonique	Opération symbolique	Drapeaux S	Z		H		P/V	N	C	Opc 76	543	210	Hex	Nombre d'octets	Nombre de cycles M	Nombre d'états T	Commentaires
DAA	Convertit le contenu acc. en BCD après addition ou soustraction avec opérandes BCD	↕	↕	X	↕	X	P	•	↕	00	100	111	27	1	1	4	Accumulateur ajust. décimal
CPL	A ← A	•	•	X	1	X	•	1	•	00	101	111	2F	1	1	4	Accumulateur de complément (compl. à 1)
NEG	A ← 0 − A	↕	↕	X	↕	X	V	1	↕	11 01	101 000	101 100	ED 44	2	2	8	Acc. négat. Complément à 2)
CCF	CY ← CY	•	•	X	X	X	•	0	↕	00	111	111	3F	1	1	4	Drapeau de retenue compl.
SCF	CY ← 1	•	•	X	0	X	•	0	1	00	110	111	37	1	1	4	
NOP	Pas d'opérande	•	•	X	•	X	•	•	•	00	000	000	00	1	1	4	Drapeau de retenue mis
HALT	Arrêt UC	•	•	X	•	X	•	•	•	01	110	110	76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11	110	011	F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11	111	011	FB	1	1	4	
IM 0	Met interrupt. mode 0	•	•	X	•	X	•	•	•	11 01	101 000	101 110	ED 46	2	2	8	
IM 1	Met interrupt. mode 1	•	•	X	•	X	•	•	•	11 01	101 010	101 110	ED 56	2	2	8	
IM 2	Met interrupt. mode 2	•	•	X	•	X	•	•	•	11 01	101 011	101 110	ED 5E	2	2	8	

Notes : IFF indique la bascule d'interruption ;
CY indique la bascule de retenue ;
* indique que les interruptions ne sont pas échantillonnées à la fin de EI ou DI.

Notation de drapeaux : • = drapeau non affecté, 0 = drapeau à zéro ;
1 = drapeau mis, X = drapeau inconnu ;
↕ = drapeau affecté par le résultat de l'opération.

Page manquante
(publicité)

Page manquante
(publicité)